2015

A neuro-genetic hybrid approach to automatic identification of plant leaves

Oluleye Hezekiah Babatunde
Edith Cowan University

A Neuro-Genetic Hybrid Approach To Automatic Identification Of Plant Leaves

By

Oluleye Hezekiah Babatunde

A dissertation for the degree of Doctor of Philosophy

(PhD) in Computer Science

Submitted to

School of Computer and Security Science

Edith Cowan University, Perth, Western Australia

Supervisors

Dr Leisa J. Armstrong

Dr Jinsong Leng

Professor Dean A. Diepeveen

October 08, 2015

Contents

Page Number

Acknowledgements

I extend all glory to Jehovah God- the Father of our Lord JESUS who gave me life, breadth, and all things. My goodness is nothing apart from HIM.

I would like to express my special appreciation and thanks to my supervisors Dr. Leisa Armstrong (didi), Dr. Jinsong Leng and Professor Dean Diepeveen. You have all been tremendous mentors for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless.

A special thanks to my family. Words cannot express how grateful I am to my wife Esther Babatunde and my three wonderful boys Daniel, David and Joseph for their patience, support and all of the sacrifices that they have made on my behalf, although I have deprived them of fatherly fellowship because of this study. Your prayers for me was what sustained me thus far. I would also like to thank all of my friends in the PhD laboratory who gave me valuable information during my study. Your proximity to my PhD desk, Sreedhar Nallan, was divine and helpful. I would like express appreciation once again to my beloved wife Esther Babatunde who spent sleepless nights (4 or 5 days a week) at work to support me financially.

Many thanks to Nigerian government and ECU for ECUIPRS and TETF scholarship respectively given to me.

My special thanks to my wonderful father and mother (Mr Moses Babatunde and Mrs Juliana Babatunde) and my siblings in Nigeria and Germany.

Declaration

I certify that to the best of my knowledge:

1. this dissertation comprises my original work;
2. due acknowledgement has otherwise been made to all other material used
3. this dissertation has been substantially accomplished during my current enrolment, and has not previously been accepted for any degree at this or another institution.

Student's Name : Babatunde Oluleye Hezekiah

Signature: ————————————————-

Date: ————————————————-

List of Abbreviations

ACH	Angle Code Histogram
ANN	Artificial Neural Network
AR	Aspect Ratio
BGLAM	Basic Grey Level Aura Matrix
CCD	Centroid Contour Distance
CNN	Cellular Neural Network
EM	Expectation Maximization
EFD	Elliptic Fourier Descriptor
FF	Form Factor
FT	Fourier Transform
FD	Fourier Descriptors
FFT	Fast Fourier Transform
GA	Genetic Algorithm
GRNN	General Regression Neural Networks
GLCM	Gray-Level Co-Occurrence Matrix
GUIDE	Graphical User Interface Development Enviroment

HA	Harmonic Analysis
IDSC	Inner Distance Shape Context
IDE	Integrated Difference Entropy
KNN	K Nearest Neighbour
LMI	Legendre Moment Invariant
MSE	Mean Square Error
MLE	Maximum Likelihood Estimator
MMC	Moved Median Centres
MOEA	Multi-Objective Evolutionary System
MLP	Multi-Layer Perceptron
NBC	Naive Bayes Classifier
NF	Narrow Factor
NGHIS	Neuro-Genetic Hybrid Intelligent System
ODE	Ordinary Differential Equation
PNN	Probabilistic Neural Network
PDF	Probability Density Function(pdf)
PCA	Principal Component Analysis
PC	Principal Component
PSO	Particle Swarm Optimization
PFT	Polar Fourier Transform

PDE	Partial Differential Equation
PLRS	Plant Leaf Recognition System
ROI	Region of Interest
RGB	Red Green Blue
RMI	Regional Moment Inertial
SPPD	Statistical Properties of Pores Distribution
SVM	Support Vector Machine
sRGB	Standard Red Green Blue
TMI	Tchebichef Moment Invariant
WT	Wavelet Transformation
ZMI	Zernike Moment Invariant

List of Tables

List of Figures

Publications

1. Journal paper 1 (Published):

   Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014). Zernike Moments and Genetic Algorithm : Tutorial and Application. British Journal of Mathematics & Computer Science, 4(15), 2217-2236.

2. Journal paper 2 (Published):

   Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014). On the application of genetic probabilistic neural network and cellular neural networks in precision agriculture. Asian Journal of Computer and Information Systems, 2(4), 90-101.

3. Journal paper 3 (Published):

   Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014). A Genetic Algorithm-Based Feature Selection. International Journal of Electronics Communication and Computer Engineering, 5(4), 889-905.

4. Journal paper 4 (Published):

   Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2015). A survey of computer-based vision systems for automatic identification of plant species. Journal of Agricultural Informatics; 6(1), 61-71.

5. Journal paper 5 (Published):

   Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2015). Comparative analysis of Genetic Algorithm and Particle Swam Optimization: An application in precision agriculture. Asian Journal of Computer and Information Systems, 3(1), 1-12.

6. Journal paper 6 (Published):

   Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2015b). A computer-based vision systems for automatic identification of plant species using knn and genetic pca. Journal of Agricultural Informatics, 6(2), 32-44.

7. Journal paper 7 (Published):

   Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2015). A Neuronal Classification System for Plant Leaves using Genetic Image Segmentation. British Journal of Mathematics & Computer Science. doi:10.9734/BJMCS/2015/14611; 9(3), 261-278.

8. Journal paper 8 (Under review):

   Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014). A Neuro-Genetic Intelligent System for Classification of Plants Species. International Journal of Pattern Recognition and Artificial Intelligence.

9. Conference paper 1 (Published):

   Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014a). Application of cellular neural networks and naivebayes classifier in agriculture. In proceedings of AFITA 2014, 9th Conference of the Asian Federation for Information Technology in Agriculture, Australia, Perth, 29 September to 2nd October 2014, 63-72.

10. Conference poster (Published):

    Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014). A computational approach to plant leaves identification. In proceedings of AFITA 2014, 9th Conference of the Asian Federation for Information Technology in Agriculture, Australia, Perth, 29 September to 2nd October 2014.

Abstract

Plants are essential for the existence of most living things on this planet. Plants are used for providing food, shelter, and medicine. The ability to identify plants is very important for several applications, including conservation of endangered plant species, rehabilitation of lands after mining activities and differentiating crop plants from weeds.

In recent times, many researchers have made attempts to develop automated plant species recognition systems. However, the current computer-based plants recognition systems have limitations as some plants are naturally complex, thus it is difficult to extract and represent their features. Further, natural differences of features within the same plant and similarities between plants of different species cause problems in classification.

This thesis developed a novel hybrid intelligent system based on a neuro-genetic model for automatic recognition of plants using leaf image analysis based on novel approach of combining several image descriptors with Cellular Neural Networks (CNN), Genetic Algorithm (GA), and Probabilistic Neural Networks (PNN) to address classification challenges in plant computer-based plant species identification using the images of plant leaves.

A GA-based feature selection module was developed to select the best of these leaf features. Particle Swam Optimization (PSO) and Principal Component Analysis (PCA) were also used sideways for comparison and to provide rigorous feature selection and analysis. Statistical analysis using ANOVA and correlation techniques confirmed the effectiveness of the GA-based and PSO-based techniques as there were no redundant features, since the subset of features

selected by both techniques correlated well. The number of principal components (PC) from the past were selected by conventional method associated with PCA. However, in this study, GA was used to select a minimum number of PC from the original PC space. This reduced computational cost with respect to time and increased the accuracy of the classifier used. The algebraic nature of the GA's fitness function ensures good performance of the GA. Furthermore, GA was also used to optimize the parameters of a CNN (CNN for image segmentation) and then uniquely combined with PNN to improve and stabilize the performance of the classification system. The CNN (being an ordinary differential equation (ODE)) was solved using Runge-Kutta 4th order algorithm in order to minimize descritisation errors associated with edge detection.

This study involved the extraction of 112 features from the images of plant species found in the Flavia dataset (publically available) using MATLAB programming environment. These features include Zernike Moments (20 ZMs), Fourier Descriptors (21 FDs), Legendre Moments (20 LMs), Hu 7 Moments (7 Hu7Ms), Texture Properties (22 TP) , Geometrical Properties (10 GP), and Colour features (12 CF). With the use of GA, only 14 features were finally selected for optimal accuracy. The PNN was genetically optimized to ensure optimal accuracy since it is not the best practise to fix the tunning parameters for the PNN arbitrarily. Two separate GA algorithms were implemented to optimize the PNN, that is, the GA provided by MATLAB Optimization Toolbox (GA1) and a separately implemented GA (GA2). The best chromosome (PNN spread) for GA1 was 0.035 with associated classification accuracy of 91.3740% while a spread value of 0.06 was obtained from GA2 giving rise to improved classification accuracy of 92.62%.

The PNN-based classifier used in this study was benchmarked against other classifiers such as Multi-layer perceptron (MLP), K Nearest Neigbhour (kNN), Naive Bayes Classifier (NBC), Radial Basis Function (RBF), Ensemble classifiers (Adaboost). The best candidate among these classifiers was the genetically optimized PNN. Some computational theoretic properties on PNN are also presented.

Chapter 1

Introduction

1.1 Overview

Plants are essential for the existence of most living things. For example, the maintenance of and the balance of oxygen and carbon dioxide are done by plants through the process of photosynthesis. Plants provide shelter, foods, and medicine for human beings. They are also fundamental piece in the puzzle to solve the problem of global warming (Whelan, 2003; McBratney, Whelan, & Ancev, 2005; Garcia & Barbedo, 2013). Unfortunately, many of these plant species are being endangered due to human activities such as grazing by animals, mining, chemical industries activities , natural disasaters such as volcanic erruptions and encroachment of forests.

The preservation of plants requires an efficient means for their identification. This has traditionally been done by taxonomists, botanists, and other professionals through the manual matching of the plant's features such as leaves, flowers, and bark or by comparing them with previously collected specimens or by using books or identification manuals (Meeta, Mrunali, Shubhada, Prajakta, & Neha, 2012; Fan, Peng, Gao, & Zhou, 2015).

The manual approach to plant classification is slow and prone to human error. As a result, using digital photographs of leaves, attempts to automate this process have been made using features of plants extracted from images as input parameters to various classifier systems (Pahalawatta, 2008;

Cope, Corney, Clark, & Remagnino, 2012; Prasad, Peddoju, & Ghosh, 2013; Kawulok & Nalepa, 2014a; Kulkarni, Rai, Jahagirdar, & Upparamani, 2014; Abdul, 2014; Charters, Wang, Chi, Tsoi, & Feng, 2014). Computer vision or image recognition systems use certain attributes (physiological or behavioural characteristics) of the objects being recognised as the basis of classification. For example, fingerprints, face, iris, speech, gait and hand geometry are the most commonly used features in human recognition systems (Pahalawatta, 2008; John, Allen, Arvind, Shankar, & Yi, 2009). The techniques employed by object classification systems generally involve pattern recognition techniques to carry out object query from a database. These systems are equipped with capabilities to either reject or accept the submitted claim or object query by comparing it with the pre-stored trained objects in the database. It also requires input from experts with specific knowledge domain (botanists). The essence of computer-based plants classification system is to augment the manual techniques so as to increase speed, efficiency, and accuracy of recognition.

1.2 Problem Statement and Motivation

There are at least 250,000 plant species recorded (Govaerts, 2001; Scotland & Wortley, 2003; Chomtip, Chawin, Pitchayuk, & Nititat, 2011; Cope et al., 2012), and there have been several attempts recently to develop computer-based classification systems for plants species (Pahalawatta, 2008; Goeau et al., 2011; Fan et al., 2015). However, plant's leaf variability within and between species, the extraction of complex plant leaf features, classification accuracy and query speed are still the major concerns in this area (Goeau et al., 2011; Arora, Gupta, Bagmar, Mishra, & Bhattacharya, 2012; Laga, Kurtek, Srivastava, Golzarian, & Miklavcic, 2012; Pundkar & Waghmare., 2014).

In other applications of artificial inteliigence (AI) research, it is often found that hybrid classifier models perform better than a single model (Tian, Hu, Ma, & Ha, 2012; Valliammal & Geethalakshmi, 2011a). The use of a hybrid system may address some of the challenges facing computer-based plants recognition systems.

1.3 Research Objectives

The objectives of this study are:

1. To extract and evaluate large sets of existing leaf features based on shape, colour, and texture (e.g. geometric, morphological, texture, and moments, etc) for classification of leaves.

2. To investigate new approach applying hybrid techniques involving CNN, PNN, and GA for feature selection and classification of plants based on the images of their leaves.

1.4 Significance of the Study

The approach in this work was based on a novel hybrid model. A CNN-PNN-GA model was developed to perform computer-based plant recognition. From a taxonomical point of view, identifying plant species is usually very difficult for both general public and even some professionals. This work will provide easy access to archives for botanists, farmers, and foresters. This study may also be used to provide discriminative measures against weeds which are often, not wanted since they impede the growth and health of wanted plants such as crops and native species.

This research combines CNN, PNN and GA with a large number of features extracted from images of plants leaves. The CNN-PNN-GA is novel application in this study area. The CNN is efficient in pattern recognition and it is extremely fast, while radial basis layer of the PNN can approximate any function and GA is known globally to be good at parameter optimization (Melanie, 1999). The PNN is also very easy to train since its weights are not trained but assigned. A hybrid model (CNN-PNN-GA) comprising of all these architectures is envisioned as a good choice for computer-based plant recognition. The existing works have some limitations such as low discriminating power between some crop plants and weeds, rejection of variability within the same species and acceptance of variability between different species of plants on plant species recognition. The proposed method in this study, may provide more opportunities for plant classification and to make use of, and preserve plants species for human and animals use. This work also serves as a framework for other tasks such plant lesion detection, forensic application and image recognition in general. The practical application of this study may be found in the following examples.

1. Botanical Achives: This study could be useful for provision of botanical information for botanists, farmers, foresters.
2. Forensic applications: This work can easily be applied in forensic investigation for identification of leaves.
3. Pattern recognition: This work could be used for pattern recognition models in bioinformatics and other real life scenarios.
4. Computational mathematics: This work investigates the use of ordinary differential equation (ODE) in image processing or computer vision. This will facilitate collaboration between applied mathematicians and image processing researchers and thus provide a future link between the two for better research outcomes.
5. Plant conservation: This work could also be useful in plant conservation to provide information for botanists researching endangered plants.
6. General image processing: This study could also provide information on low-level image processing which can facilate further research in the field of image processing.

1.5 Research Question

How can a hybrid-based approach based on CNN, PNN, and GA be employed for plant leaf classification systems?

1.5.1 Sub-Research Question 1

What are the suitable techniques for image segmentation for feature extraction in plant leaf classification systems?

1.5.2 Sub-Research Question 2

How can GA be used to obtain suitable set of plant leaf features (shape, colour, and texture) associated with plant leaf recognition?

1.5.3 Sub-Research Question 3

How can effectiveness of GA compared to PSO, and PCA be established for feature selection?

1.5.4 Sub-Research Question 4

How can PNN-based classification of leaves be optimized through the use of GA techniques?

1.6 Key Contributions of the Study

The major contribution of this study is the development of a hybrid system that includes the image feature analysis, some optimization techniques, and the efficient discriminative models , with the application to the plant species classification using the images of plant leaves. This study particularly examines a variety of techniques such as several image features, genetic algorithm (GA), particle swarm optimization (PSO), principal component analysis (PCA), cellular neural networks (CNN), probabilistic neural networks (PNN) and numerical solutions of ordinary differential equations (ODE) in order to improve on the current computer-based vision systems for plant classification.

The key contributions of this study are itemized as follows:

1. Extraction of large numbers of image descriptors. Most of those descriptors are about affine maps (descriptors invariant to scaling, rotation and translation). This is a novel amalgation of several image features in one study. This eliminates some of the misclassification issues in the current systems. (see Chapter 4).

2. Provision of GA, PSO and PCA-based feature selection (see Chapters 5 & 6). The PCA

in this study was GA-based, a unique idea in pattern recognition. The number of principal components was genetically selected instead of manual reduction. This is a novel approach in feature analysis.

3. Rigorous feature analysis (see Chapters 5 & 6). In Chapter 6, the features selected by both GA and PSO were further subjected to statistical tests (ANOVA and correlation) to ascertain that there were no redundant features among them. ANOVA analysis of GA-based and PSO-based features is novel idea.

4. Unique and detailed description of both theoretical and numerical properties of PNN (see Chapter 7).

5. Parameter Optmizations for PNN and CNN (see Chapter 8 and Figure 4.4).

6. Unique image classification systems for the practical applications in agriculture (see Figures 7.8 & 7.9).

1.7 Thesis Organization

This thesis is organized as follows;

Chapter 1 provides the background information, research questions, aims and objectives of the study, and the significance of the research.

Chapter 2 provides a review of literature of previous research on the area of manual leaf classification, area of plant recognition systems, an overview of techniques for image classification systems which includes pre-processing, segmentation, features extraction, features selection and image classification.

Chapter 3 provides an outline of research methodology used in this research. It includes sections on research methodology, proposed research methodology, research approach and acitivies, information on datasets and computational platforms used in this study.

Chapter 4 provides details of the results of image pre-processing, segmentation and feature extraction from the leaf data set and a summary of the feature set.

Chapter 5 provides details implementation and the results of the GA and the GA-based feature selection techniques employed on the dataset. It details the dataset used, the implementation of the GA used, simulation and experimental results, validation of results and conclusions.

Chapter 6 provides a comparison of the results of GA feature selection approach compared to particle swarm optimization (PSO) and principal component analysis (PCA). The implementation details of both PSO and PCA are also detailed in this chapter.

Chapter 7 provides the results of classification of dataset using PNN classification approach. The theoretical, computational properties and general overview of PNN are presented in this chapter. The design of the classification system is also detailed in this chapter. This chapter also outlines the experimental validation, dataset and results.

Chapter 8 provides further details about the application of GA to optmize the PNN classification approach.

Chapter 9 provides the general discussion of findings and conlusions. It also details proposed future works in relation to this thesis.

Chapter 2

Literature Review

2.1 Introduction

Traditional recognition of plant species is carried out by manual matching of the plant's features, relating to components of the plant, such as leaves, flowers, and bark, against a taxonomical atlas (Meeta et al., 2012). Attempts to automate this process have been made, using features of plants extracted from images as input parameters to various classifier systems (Cope et al., 2012). Since plant leaves are often more available than the fruits and flowers, and because leaves are also mostly two-dimensional (2D) in shape, most of the existing work on computer-based plant recognition are based on the leaves of plants.

This chapter provides a review of the techniques used in the field of automated plant recognition as well as the techniques that will be employed in the proposed research. Section 2.2 introduces the leaf categories used in manual species recognition. Section 2.3 details some geometrical and morphological properties of images of plant leaf. Section 2.4 provides overview of image classification systems followed by feature selection techniques followed by general summary. section 2.5 provides a review of computer-based plant recognition systems. Most contents of this chapter has been published in the paper (O. Babatunde, Armstrong, Leng, & Diepeveen, 2015c).

2.2 Leaf Characteristics in Manual Identification

The shape of a leaf is an important feature of plant development that depends on genetic, hormonal and environmental factors (Weight, Parnham, & Waites, 2008). The shape and structure of leaves often vary from species to species of plant depending on the adaptibility to climatic conditions and as well as availability of light. As seen in Figure 2.1a, a normal leaf of an angiosperm consists of a petiole (leaf stalk), a lamina (leaf blade), and stipules (small structures located to either side of the base of the petiole). According to Pat (2000), leaves can be categorized in many ways. For instance, a leaf can be classified as either broad or narrow. A broad leaf has a wide blade, having a visible vein alignment, as in the Northern Catalpa, shown in Figure 2.1(a). Slender leaves on the other hand have narrow, needle-like leaves, as with the Norway spruce, shown in Figure 2.1(b). The full range of leaf categories documented by Pat (2000), is reproduced in Table 2.1. Information about plants' numenclature can be obtained from http://oregonstate.edu/dept/ldplants/Plant%20ID-Leaves.htm. Cope et al. (2012) asserts that the most discriminative feature of a plant's leaf is its shape. Some other shapes and types of leaves are shown in Figure 2.1c.

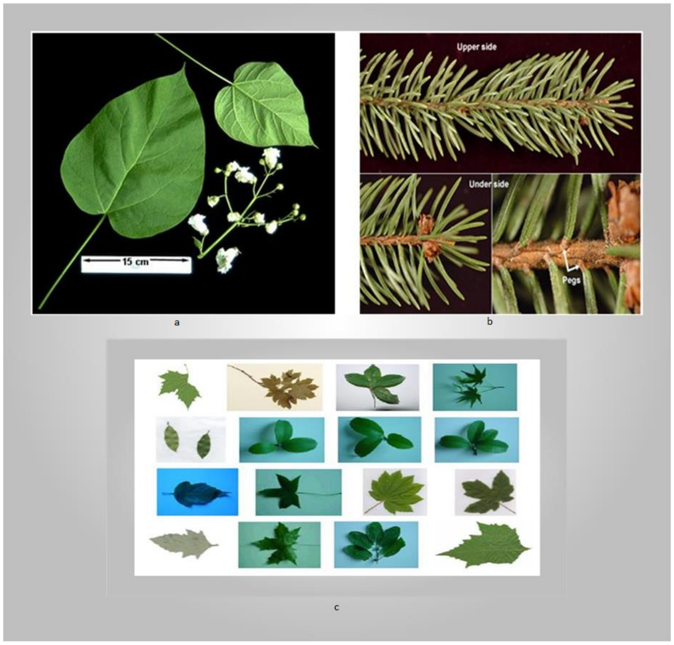


Figure 2.1: Broad Leaf Image of Northern Catalpa, Narrow Leaf Image of Norway spruce, and sample leaves with different shapes taken from (Pat, 2000, Ji-Xiang, 2005). Particular references to this figure is given as Figure 2.1a , Figure 2.1b, & Figure 2.1c from top (left to right) and then bottom.

Table 2.1: Categorizing Leaves by their shape and structure (Pat, 2000)

Leaf Type	Definition	Example
Broad	Leaf with wide blade, often with visible network of veins	Northern Catala
Alternate	Slender leaf without a wide blade. Often called needle or scale-like	Norway Spruce
Opposite	Two leaves on the same stem but in opposite direction	Common Boxwood
Whorled	More than two leaves from the same location on a twig	Redvein Enkianthus
Simple	Have only one blade divided into parts	White Alder
Compound	More than one blade and may have a complex leaf stalk structure	Paperbark Maple
Palmate	Have three or more leaflets attached at the end of stalk(petiole)	Horsechestnut
Pinnate	Have a number of leavelet attached along a central stalk	American Yellowood
Lobed	Have a curved or rounded projection	Hedge Maple
Unlobed	Doesn't have any curved or rounded projection	Western Catalpa
Entire	Have smooth edges or small notches or teeth along the margin	White Forsythia
Toothed	Have teeth at the base , at the tip, or along margin	Paperbark Maple
Clusters	At least 5 leaves together	Deoder Cedar

2.3 Geometric and Morphological Features of Leaves

The common geometric and morphological features which are associated with and have been extracted from images of plant leaf are diameter, physiological length and width, area , aspect ratio, circularity, solidity, convexity, form factor hydraulic radius, and irregularity (Wu et al., 2007; Nixon & Aguado, 2002; Kadir, 2011; Russ, 2011; Abdul, Lukito, Adhi, & Santosa, 2012; Prasad et al., 2013). These features were used by Wu et al. (2007) for identification of plant species. Their main advantage is their invariance to rotation but they are not invariant to scaling and translation. For example, Figure 2.2 shows a typical physiological length and width of an image of a plant leaf.

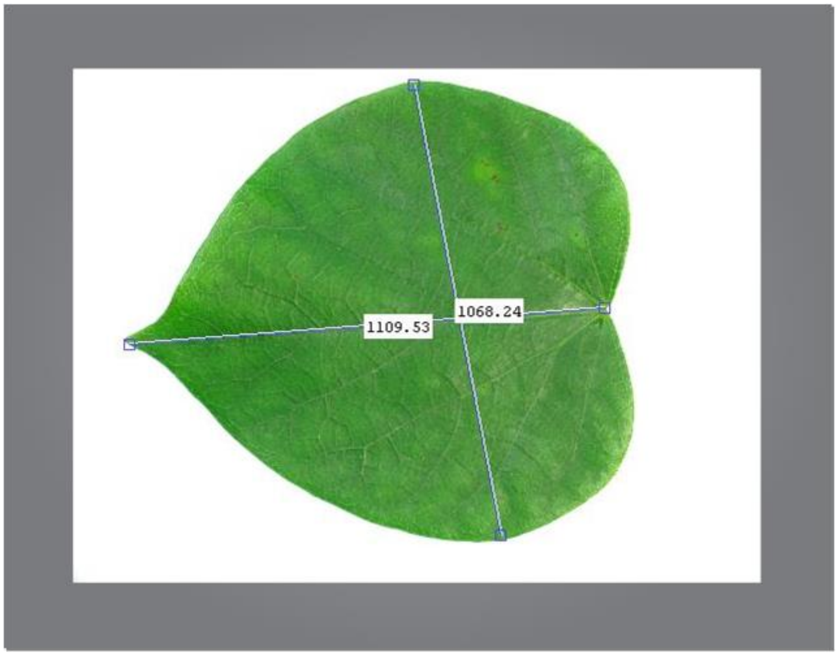


Figure 2.2: Image showing the length and width of a leaf

2.4 Overview of Image Classification Systems

The typical stages used in a computer-based plant species recognition system are shown in Figure 2.3. Going from a physical leaf to knowing its species involves steps such as image acquisition, pre-processing, segmentation, feature extraction and classification. Each of these steps will be reviewed in the following subsections.

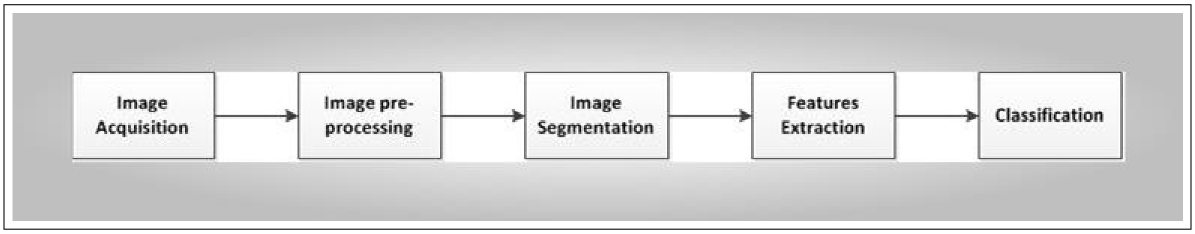


Figure 2.3: Conceptual Diagram for Image Classification System

2.4.1 Image pre-processing

Data pre-processing or image pre-processing is often required in pattern recognition systems. In image classification systems, examples of image pre-processing are image resizing and colour-to-gray scale conversion. A grayscale digital image is an image with a single sample pixel value, having only intensity information. Converting a colour image to grayscale involves mapping the multiple colour channels to a single grayscale value, usually as a weighted sum. There are several formulas for converting colour images to grayscale depending on the weighting factors of the colour channels (Christopher & Garrison, 2012). A very popular formula for colour to grayscale conversion, related to luminance properties, and the standard algorithm used by many image processing packages (including MATLAB) is shown in Equation 2.4.1.

$$\boldsymbol{Grayscale}1 = 0.2989\boldsymbol{R} + 0.5870\boldsymbol{G} + 0.1140\boldsymbol{B} \tag{2.4.1}$$

The R, G, and B correspond to the red, green and blue colour of the pixel, respectively, while their coefficients represent human perception of the colour. According to Aliaga (2010), the retina of human eyes has three types of cone which are L-cone (most sensitive to red light), M-cone (most sensitive to green light), and S-cone (most sensitive to blue light). The formula in Equation 2.4.1 was derived to mimic human-based perception brightness based on M-cone of human eyes retina. The simplest colour-to-grayscale algorithm, which is the mean of the RGB channels, is given in Equation 2.4.2.

$$\boldsymbol{Grayscale}2 = \frac{\boldsymbol{R}+\boldsymbol{G}+\boldsymbol{B}}{3} = 0.3333\boldsymbol{R} + 0.3333\boldsymbol{G} + 0.3333\boldsymbol{B} \tag{2.4.2}$$

Christopher and Garrison (2012) described thirteen methods having linear order in time complexity for converting a digital image from colour to grayscale. Each of the functions assumes the form:

$$F_{Color2Gray} := \ \mathrm{R}^{n\times m\times 3} \rightarrow \mathrm{R}^{n\times m} \tag{2.4.3}$$

where n and m are the pixel dimensions, $\mathrm{F}_{Color2Gray}$ is the formula for the conversion, $\mathrm{R}^{n\times m\times 3}$ is the colour image and $\mathrm{R}^{n\times m}$ is the grayscale image. In the combined analysis, using mean rank performance of each grayscale, Christopher and Garrison (2012) stated that the simplest method (Equation 2.4.2) with prior gama correction performed best. According to Stokes, Anderson, Chandrasekar, and Motta (1996) , the gamma expansion (with a gamma of 2.2) used by most methods in linear intensity encoding for RGB (Red Green Blue) colour space is:

$$Color_{linear} = \begin{cases} \frac{Color_{sRGB}}{12.92} & \text{if } Color_{sRGB} \leq 0.04045, \\ \frac{(Color_{sRGB}+0.055)^{2.4}}{1.055} & \text{if } Color_{sRGB} \geq 0.04045 \end{cases} \tag{2.4.4}$$

2.4.2 Image Segmentation Techniques

Image segmentation is a process of of partitioning a given image into region of interests (ROIs). The ROIs are homogenous groups such that each region is homogenous but the union of two non adjacent regions also homogenous. Awad (2009) applied variable hybrid genetic algorithm (GA) for segmentation on satellite images. Image segmentation may involve image processing techniques such as extraction of edge pixes and separation of foreground image from background image. Image segmentation are globally categorized into (i) a region-based segmentation and (ii) a boundary-based segmentation. According to Valliammal & Geethalakshmi (2011) , segmentation techniques can be typified into histogram-based techniques (Brendo, Nikola, & Howard, 2011), cluster-based techniques, compression-based techniques, edge detection, PDE-based methods, region growing methods, graph partitioning methods, watershed transformation, and curve evolution methods.

Conventional edge detection techniques have been widely used by many researchers implementing computer-based plant classification systems. (Kenji & Morio, 1984; Russ, 2011; Abdul et al., 2012). An edge is defined as a pixel at which the image values undergo a sharp variation – pixels with large element (Hezekiah, Akinwale, & Folorunso, 2010; Goto, Hirano, & Sakurai, 2014). Edge detection was applied in the measurement of plant stomata aperture by Kenji and Morio (1984).

Examples of edge detection techniques include Sobel (Sobel, 1970), LoG, Canny (Canny, 1986), Prewitt (Prewitt & Mendelsohn, 1966), and cellular neural networks (CNN) with edge detection templates for binary images(Chua & Roska, 2002). CNN is considered by many people to be the most appropriate for edge detection due to its fast operational speed, programmability, and wide range applications (Hanggi & Moschytz, 1997; Balya & Roska, 1999; Mariofanna, Paulo, Almeida, Jun, & Marcelo, 1999; Luigi, Paolo, David, & Akos, 2001; Gilli et al., 2002; Bucolo, Caponetto, Fortuna, & Frasca, 2005; Amran & Prema, 2008; Hezekiah et al., 2010).

2.4.3 Feature Extraction Techniques

There are several ways through which images are annotated in computer systems. An annotation involves using a set of descriptors or numbers to represent the images stored in the database for effective retrieval or identification. The goal of searching for images or objects from a large dataset is shared both by scientists (developers) and general users. To this end, image features can be categorized into (1) shape features, (2) texture features, and (3) colour features.

2.4.3.1 Shape Features

A: Zernike Moment

A moment describe the layout (arrangement of image pixels). Moments are global region-based descriptors for shape and is bit like combination of area, compactness, irregularity, and higher order descriptors together (Valliammal & Geethalakshmi, 2011a; Nixon & Aguado, 2012). An image moment is defined as the integration of an image function with a region-defined polynomial basis (Flusser, 2000) and (Flusser, Suk, & Zitova, 2009). The region here is defined as the area where that image is valid. From Simon(1993), the general moment M_{pq} of any image $f(x,y)$ of order $p+q$, where $p>0, q>0$, is defined as:

$$M_{pq} = \int\int_{D} pol_{pq}(x,y) f(x,y) dxdy \tag{2.4.5}$$

where $pol_{pq}(x,y)$, $i=1(0)p, j=1(1)q$ are polynomials basis functions defined on domain D. The Zernike moment (ZM) can be defined as a set of complete complex orthogonal basis functions that are square integrable and that are defined over the unit disk. An open disk around a given point, say, x in a plane is the set of all points in the plane whose distance from x is less than 1 (see Equation 4.1.4). ZM were first applied in image analysis for the first time by Teague (1980). ZM are orthogonal moments based on Zernike polynomials. Orthogonality referred to here means that there is no redudancy or overlapping of information between the moments. Thus moments are uniquely quantified based on their orders (Noll, 1976; Thawar, Zyad Shaaban, & Sami, 2009). The distinguishing feature of ZM is the invariance of its magnitude with respect to rotation (Vorobyov, 2011).

B: Fourier Descriptors (FDs))

In signal and image processing, any real signal has frequency domain representation. Fourier Descriptors (FD) are shape-based features that employs the application of Fourier theory to shape analysis or description (Nixon & Aguado, 2012; Kadir, 2015). The convolution operation used by Fourier descriptors is the Fourier transform (FT), which is a mathematical representation of a signal or digital image (say plant's leaf image) as a summation of complex exponentials containing varying magnitudes, frequencies, and phases (R. C. Gonzalez, Woods, & Eddins, 2009). The idea behind FT in image processing is to characterise a contour by a set of scalars which are representing the frequency content of the whole shape. FA is a subset of harmonic analysis which converts the boundary points of a shape to a function of the form of radius (angle) or $\rho(\phi)$. A radius is normally drawn from the centroid as a function of angle and then plotted to unroll the shape of the given image. This function is periodic in 2π and allows for the determination of the constants a and b in the series expansion given shown in Equation 2.4.6.

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n cos(\frac{2\pi nx}{L}) + b_n sin(\frac{2\pi nx}{L})\right) \tag{2.4.6}$$

Thus, as documented in Mathworks (2009), FA is an off-shoot of Spectral Analysis (a process of identifying component frequencies in a given data or signal). Frequency domain is attractive for image processing in that it makes large filtering operations run faster. Fourier transform provides easy navigation both forward and backward direction from spatial domain to frequency domain. Unlike the Zernike polynomials which are orthonomal over circular pupils, Fourier series are also orthonomal, but over rectangular pupils, and not circular pupils.

2.4.3.2 Colour features

In most image recognition systems, there often arises the need to use color features and or convert color images to grayscale images for computational simplicity (Cerutti, Tougne, Mille, Vacavant, & Coquin, 2013; Goto et al., 2014). A number of color spaces have been widely used in literature, such as RGB, LUV, HSI, HSV, and HMMD (Ping Tian, 2013). These colors, being perceived by human, and as reported by Nixon and Aguado (2012), can be derived from the amalgamation of three primary colors (three-band monochrome image data). The space for these colors is linear since summations are being used for the amalgamation of the colors. The actual information stored in any digital image is a function of the brightness information in each {R,G,B} spectral band (Umbaugh, 2011). Whenever the image is displayed, the associated brightness information is displayed on the screen by picture elements (pixel) that emit light energy corresponding to that particular color. It is therefore useful to know that the representation of these colors is based on the relationships between colored light and perception. Grayscale images on the other hand, use a single value per pixel that is called intensity or brightness. This intensity represents the amount of light reflected or emitted by an object and is dependent on the object's material properties as well as on the sensitivity of the camera sensors. According to the tristimulus theory (Nixon & Aguado, 2012), all possible colors perceived by human can be defined in a 3D linear space. Given that the weight (components) for the primary colors are W_r, W_g, W_b, then the colormetric equation for the

Red, Green, and Blue components for a color image is given as

$$C = W_r R + W_g G + W_b B \tag{2.4.7}$$

The color features are extracted once the color space is identified. The common color features that have been used in literatures are color histogram, color moments, color coherence vector (CCV), and color moments. Color moments is one of the simplest to use and yet, they are very effective features (Flicker, Sawhney, & Niblack, 1996). In relation to feature extraction, Simon (1993) stated that the colour moments of order $p+q$ and degree $a+b+c$, can be given as Equation 2.4.8

$$M_{pq}^{abc} = \int\int x^p y^q R(x,y)^a G(x,y)^b B(x,y)^c dxdy \tag{2.4.8}$$

where R, G, and B are Red, Green, and Blue color components over the pixels (x,y) and a,b,c,p, and q are positive integers. The popular color moments (CMs) are mean, standard deviation, and skewness. The moments of colour were used for leaf classification in (Kadir, 2011). The CMs have been found to be very simple to calculate and easily extracted. They are also compact and robust to use. The CCV involves high dimension and high computational cost. The strength and weakness of some color features have been reported in (Ping Tian, 2013). The CMs were reported in this paper to be simple and effective.

2.4.4 Feature Selection Techniques

High dimensional feature set could pose a great difficulty to pattern or image recognition systems. This threat is known as the curse of dimensionality (Richard & Stuart, 1962; Leng, Valli, & Armstrong, 2010). In other words, too many features often require intensive computation and also reduce the classification accuracy of the recognition system since some of the features may be redundant and non-informative (Richard & Stuart, 1962; Maldonado & Weber, 2009; Bruzzone & Persello, 2010; He, Fataliyev, & Wang, 2013; Pookhao et al., 2015). Different combinatorial set of features should be obtained in order to keep the best combination to achieve optimal accuracy. In machine learning and statistics, feature selection, which is also called variable or attribute selection, is the process of obtaining a subset of relevant features (probably optimal) for the purpose of constructing machine models (MathWorks, 2013; O. Babatunde, Armstrong, Leng, & Diepeveen, 2014b). The different feature selection methods are generally categorised into filter, wrapper and embedded method.

2.4.4.1 Filtering-based approach

In filtering algorithms, some evaluation functions are needed to be used independently of the classifier in selecting the feature subsets. Examples of filtering algorithms are Chi squared test, information gain and correlation coefficient scores. These methods use metric measure, information gain, dependency and consistency. The simplest of this methods is best individual features, where a function is used to rank individual features and the highest ranked p features are selected. The low scoring features are removed. Somol, Baesens, Pudil, and Vanthienen (2005) applied filter-based selection algorithm on several real-world datasets (e.g credit scoring dataset) using different types of classifiers. The paper ascertained that filter-based algorithms employ both the classifier-related and probabilistic-related computation and can be more vulnerable in the presence of numerical problems. The major advantages of this method is that they are computationally simple and fast and they are also carried out independently of the classification algorithm (Y. Wang, Fan, & Cai, 2014).

2.4.4.2 Wrapper-based approach

In the wrapper-based approach, feature subset selection is done through evaluation of each candidate subset with estimation obtained from the classifier or learning algorithm (Cordon, Herrera, DelJesus, & Villar, 2001; Sattiraju, Manikantan, & Ramachandran, 2013; Shanmugapriya & Padmavathi, 2013; O. Babatunde et al., 2014b; Gromski et al., 2014; Pookhao et al., 2015). Wrapper algorithms interact with the learning algorithm and model feature dependencies. However, the effectiveness of FS is dependent on the classifiers being selected. Leng et al. (2010) applied wrapper-based feature selection algorithm, genetic algorithm (GA) and k nearest neighbor (kNN) on some dataset for ranking the importance of the available features. The authors stated in the paper that wrapper-based feature selection was good for the analysis of data structure and removal of noisy and irrelevant features in large data sets which can subsequently improve the performence of various classifiers. An example of wrapper-based algorithm is recursive feature elimination algorithm. Wrapper-based FS are often used by many researchers as it guarantees better accuracies (Somol et al., 2005). Wrapper-based feature selection based on Adaptive Multi-Level Threshold Binary Particle Swarm Optimization (ABPSO) was used by Gromski et al. (2014) to search the feature space for the optimal feature subset in building face recognition system under varying background. A significant increase in the classification accuracy and substantial reduced feature set were observed.

2.4.4.3 Embedded hybrid approach

In this method, the search function is built into the learning or classification algorithms (Kohavi & John, 1996; Kittler, 1978). That means classifier is seen as a composite functional. In other words, the feature space is fed into the classifier and the feature selector component part of the classifier is invoked first before the classification is done. Maldonado and Weber (2011) reported the state-of-the-art on embedded feature selection using the classification method Support Vector Machine (SVM). The embedded methods involve two main steps which are (1) ranking of features from a training set and (2) classifiers-based feature selection using crossvalidation or other feature partition algorithms. The most popular examples of embedded feature selection methods are regularization methods (also called regularization methods) since they introduce additional constraints into the optimization of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (less coefficients). Three examples of regularization algorithms are the LASSO, elastic net and ridge regression. The main disadvantage of embedded feature selection is the computation time requirements and sometimes, overfitting, especially when the ranking of the features is performed using non-linear functions (Lal, Chapelle, Weston, & Elisseeff, 2006; Z. Xiao, Dellandrea, Dou, & Chen, 2008; S. Wang, Jiliang, & Huan, 2015).

2.4.5 Classification

Image classification is an area of Machine Learning (ML) which is the process of learning a set of rules or patterns from the given historical instances (or the training set) in order to predict or extract the class information of a new instance (Kotsiantis, Zaharakis, & Pintelas, 2007; Anagnostopoulos, Anagnostopoulos, Loumos, & Kayafas, 2006). Classification involves class separability measures. Image classification is the last stage in computer-based plant recognition systems. The output from the features extraction modules are used as input to train classifiers to classify plants species. The choice of the classifier is up to the developer to decide. Some typical examples of classification techniques that have been used in the past are PNN (Wu et al., 2007), SVM (Honfei, 2010; Kawulok & Nalepa, 2014b; Quadri & Sirshar, 2015), RBPNN (Gu, 2005), GRNN (Gu, 2005) , k-means (Chomtip, Supolgaj, Piyawan, & Chutpong, 2011), kNN (Gu, 2005; Aldea, Fira, & Lazar, 2014), IDSC (Ling & Jacobs, 2007; Belhumeur & David, 2011), random forests (Arora et al., 2012), Angle Code Histogram (ACH); (Z. Wang, Chi, & Feng, 2003), MMC Hypersphere (Du, 2007). A comparative analysis of different classification techniques such as ANN, SVM, Fuzzy Measures, GA and their combination were used for plants classification by Seetha, Muralikrishna, Deekshatulu, Malleswari, and Nagaratna (2008). The authors stated that the GA-ANN based model outperformed gradient descent-based neural network and other classifiers.

2.5 Computer-based plant recognition systems

2.5.1 Introduction

Several plant species recognition systems have been developed based on various features and classifiers. This section provides a summary of research reported in the literature, along with analysis of its effieciency.

2.5.1.1 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN) have been used as machine learning models in recent times due to their adaptibility and scalability. Any ANN model depends on input, activation functions, network topology and connection weights.

One study by Zalikha et al. (2011) reported one the use of PNN compared the effectiveness of Zernike Moment Invariant (ZMI), Legendre Moment Invariant (LMI), and Tchebichef Moment Invariant (TMI) as descriptor features of leaves. The data set consisted of images of 10 different plant species, with different sized leaves. Using grayscale conversion followed by thresholding, the images were converted into binary images, from which the descriptors could be derived. Scaling and rotation of the images was used to produce many variants of the images at different sizes and orientation. The incorporation of variant images allowed the system to be tested for rotation and scale invariance. A Generalized Regression Neural Network (GRNN) was used for the classification, with classification results showing that features from the TMI were the most effective. To improve this work further, more features and optimization of GRNN should be incorporated.

A computer-based vision system for identifying plants involving the use of moments as features was reported by Abdul et al. (2012), where Zernike moments were combined with geometric features, color moments and gray-level co-occurrence matrices (GLCM). The classifier used was Probabilistic Neural Networks (PNN) while the Euclidean distance was used to measure the similarity index of the leaf of query (vector 1) to every leaf in the database (vector 2). The investigation showed that Zernike moment performs better when they are combined with other features in leaf classification systems. An optimum accuracy of 94.69% was reported by using Zernike moments of order 8.

Wu et al. (2007) also applied the PNN for plant leaf classification, attempting to differentiate between 32 different plants in Yangtze, China. Twelve features (geometrical and morphological) were used. These features were: diameter, length, width, area, perimeter, smooth factor, aspect ratio, form factor, rectangularity, narrow factor, perimeter to diameter ratio, and length to width ratio. Principal Component Analysis (PCA) was used to reduce the feature vector to 5 principal components. In this project (Flavia project), the classifier (PNN) was trained using 1800 leaves. Ten leaves sample were taken from each plant, implying that the test data set contained 320 leaves. The average accuracy was recorded to be 90.31%. A major comment about this work is that the features used are not enough to ensure improved accuracy across large dataset. Moments and colour features also should be included. The parameter(s) of PNN also need to be optimized to improve the underlying learning model.

The work of Kadir (2011) involved the use of the Polar Fourier Transform (PFT) and three geometric features to represent shapes of leaves. Color moments consisting of the mean, standard deviation, and skewness were computed to represent color features. Texture features were also extracted from Grey-Level Co-occurrence Matrix (GLCM) by counting the co-occurrence pixels with grey value i and j at the given Euclidean distance . The classifier scheme used was Probabilistic Neural Networks (PNN). In testing this system, two dataset (Foliage and Flavia, see (Wu et al., 2007), were used in comparing the proposed method with the work of (Wu et al., 2007). The overall classification result was stated by the author to be 94.69%.

A hybrid approach involving a combination of Wavelet Transform (WT) and Gaussian Interpolation was proposed together with kNN and Radial Basis Probabilistic Neural Network (RBPNN) for leaves recognition by (Xiao, Ji-Xiang, & Xiao-Feng, 2005). Following image acquisition, the image was converted to greyscale and decomposed by the WT. The essence of decomposition by WT and Gaussian Interpolation was to produce low-resolution images and a series of detailed images. The wavelet features extracted by WT and Gaussian Interpolation were then used to train the kNN and RBPNN for classification. The reported accuracy in this work was 95%. The classifier (RBPNN) used in this work should be optimized. The features in this study were considered not to be adequate in number. Large features should be obtained and combinatorially selected to ensure improved accuracy.

Another approach using fuzzy selection technique based on morphological features was reported by Panagiotis (2005). After the image capture and image preprocessing, a parameterized thresholding depending on the lighting conditions was performed, followed by calculation of the centre of gravity of the leaf's image. Next, the image of the leaf is rotated to have vertical orientation. Morphological and geometrical features such as diameter, length, width, perimeter, area, aspect ratio, smooth factor, form factor, rectangularity, narrow factor, perimeter to diameter ratio, length to width ratio, and vein features were then computed. A fuzzy surface model was finally used to select images from feature database before they were fed into the RBPNN for classification. It was found that the proposed system was able to correctly classifying even deformed leaves. This paper did not state the actual quantified classification results and had limited feature set.

Jyotismita and Ranjan (2011) combined a thresholding method with H-Maxima transformation (Gonzalez, 2007) to extract veins of 180 leaves taken from a website source (Jyotismita & Ranjan, 2011). The data set was divided into three classes, Pittosporum Tobira, Betula Pendula and Cercis Siliquastrum, each consisting of 60 images. Moment-Invariants (Geometric Moments) and centroid-radii approaches were then used to extract features needed for classification. The first four normalized central moments M1, M2, M3, M4 of each image of the trained and test datasets were computed and individual features from (any of M1, M2, M3, M4) and (combinations of features from M1, M2, M3, M4) were fed into multilayer perceptron (MLP) to find the best combinations. The 180 dataset was divided into two parts where 90 images were used as training dataset (T) and the remaining 90 images as the test dataset (S). For the computation of recognition rates, comparisons between training and test datasets were done using a MLP with feed-forward back-propagation architecture which gave Mean Squared Error (MSE) of 0.005 and reached convergence in 38280 epochs. The results showed that individual moment values M1 provided the best results of 88.90%. The feature combinations M1-M3 and M1-M3-M4 provided classification results of 95.50% and 93.30% respectively. This work could be improved by using more features and optimising the ANN classifier.

Rashad, El-Desouky, and Khawasik (2011) used a combined classifier consisting of Learning vector Quantization (LVQ) and Radial Basis Function (RBF) for plant classification based on the characterization of texture properties. A digital camera was used to capture plant's images at 128 × 128 resolution. The acquired RGB images were then converted into grayscale images. Texture features were extracted from the grayscale images and using random sample of 30 blocks of each texture as a training set, and another 30 blocks as a test dataset, it was shown that the combined classifier method outperformed other methods (PCA, k-NN, RBPNN), with the least MSE and accuracy of 98.70%.

2.5.1.2 Statistical Techniques

Statistical techniques unlike the ANN, rely on the underlying probabilistic model by providing a probability that an instance belongs in each class, rather than just a classification. Bayesian networks, Naive Bayes model, and even PNN are categorized as statistical learning algorithms (Kotsiantis, Zaharakis, & Pintelas, 2006).

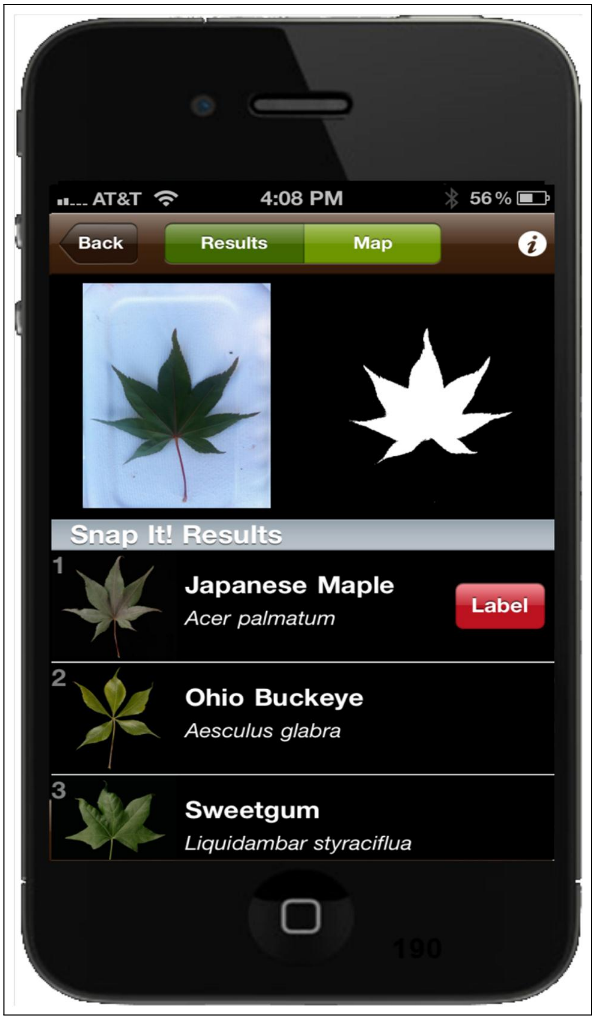
Gebhardt (2006) developed a digital image processing system for identification of broad-leaved dock (Rumex obtusifolius L.) for grass lands. The authors of this study focussed on the identification of one of the most invasive and persistent weed species on European grass lands. The total image samples used were 108 digital photographs obtained through a field experiment under constrained enviroment (i.e constant recording geometry and illumination conditions). Image segmentation in this work was done through transformation of the $\{R, G, B\}$ components of the colored images to grayscale images. Binary images were then derived from the grayscale images by applying local homogeneity threshold of value 0.97. Following this, morphological opening was performed. The features extracted were shape, color and texture-based. The learning system was based on maximum-likelihood estimation (MLE) . Furthermore, rank analysis was used for feature analysis to obtain optimal classification accuracy. The accuracy x reported with the given training set was in the range $71 \leq x \leq 95$.

A system called LeafSnap was developed by Kumar et al. (2011) for identifying tree species using the photographed images of their leaves (see Figure 2.4). The image database consists of 5972 images taken from 184 trees in the Northeastern United States. There were no needs for any color-to-grayscale conversion in this work since color segmentation was used by estimating foreground and background color distributions. The segmentation problem was solved using Expectation-Maximization (EM). The images were then resized into 300 x 400 and rotated by 90 degrees. After this, the histograms of curvatures along the contour of the leaves at multiple scales were extracted from the images of the leaves and finally, species matching was performed through 1-nearest neighbor classification. The classification accuracy as reported in this paper was 96.8%. The recognition engine of the LeafSnap consists of a backend server which accepts input images from various front-end clients. There is currently, a front-end application of LeafSnap for the iPhone and iPad devices.

2.5.1.3 Instance-based techniques

Instance-based learning models are lazy-learning algorithms which require less computational time and are based on metric between test samples and all observations in the training set. nearest neighbour algorithm (e.g kNN) is one of the most straightforward instance-based learning algorithms (Aha, 1997; De Mantras & Armengol, 1998).

In a study carried out by Sandeep and Parveen (2012), leaf color, area and edge features were used for identification of Indian medicinal plants (Hibiscus, Betle, Ocimum, Murraya, Leucas, Vinca, Ruta, Centella, Mentha). The method described in this work involved reading the test image and comparing with the database images. The images were segmented through grayscale conversion followed by binarization via thresholding and comparison of edge histogram, colour histogram, and difference in area of test and database image were carried out between a candidate image and those in the database. The candidate image was classified based on the class



Snap It! Results

1 Japanese Maple
Acer palmatum

2 Ohio Buckeye
Aesculus glabra

3 Sweetgum
Liquidambar styraciflua

Figure 2.4: LeafSnap project by (Kumar et al., 2001)

of database image it was closest to using Euclidean distance. Results showed all the plants were correctly classified except Tulsi menthe species which was wrongly identified as mint ocimum and vice-versa due to similarities in leaves veination. This work needs more discriminating features such as Zernike momemnts, Hu7 moments and some others.

Chomtip, Supolgaj, et al. (2011) developed the Thai Herb Leaf Image Recognition System (THLIRS) using k-Nearest Neighbor (kNN) as the classifier. A digital camera was first used to take the pictures of leaves, together with a one-baht coin as a size gage, against a white background. The second phase in THLIRS involved image pre-processing and segmentation (resizing, black-and-white conversion (grayscale conversion followed by thresholding), image enhancement, juxtaposition of photographed images of leaf and one-baht coin for the purpose of comparison, cropping of leaf image, and boundary tracking). The discriminative measure in the leaf-coin images on the background is that the leaf's image was assumed to be the largest object in the image, while the coin is the second largest object in the same image. In the third stage, 13 features (leaf and coin ratio, aspect ratio, roundness, ripples counting, ripples pixels counting, half-leaf area ratio, upper leaf area ratio, lower leaf area ratio, colour features, vein features (at threshold of 0.05, 0.03, and 0.01 respectively) were extracted. The dataset in THLIRS was divided into training and testing data. With a value k = 6 in the k-NN classifier, THLIRS achieved classification accuracy of 93.29%, 5.18%, and 1.53% for match, mismatch, and unknown, respectively for the training dataset, while that of test dataset was 0%, 23.33%, and 76.67% for match, mismatch, and unknown, respectively. kNN is a good classifier but not sufficient enough across large image dataset. Moment features may be included in this work to improve the overall accuracy of the system

Another study by Belhumeur et al. (2008) reported the development of a working computer vision system for identification of plant species. The e-botany (database of leaves) was made from (a) the flora of Plummers Island containing 5,013 leaves of 157 species, (b) Woody Plants of Baltimore-Washington DC containing 7,481 leaves of 245 species, and (c) Trees of Central

Park containing 4, 320 leaves of 144 species. From this collection all the images were cropped and later converted to binary images through grayscale conversion followed by thresholding. Shape distances were computed from the binary images using chi-square, while shape matching (classification) was done via Inner Distance Shape Context (IDSC). The purpose of IDSC was to retrieve coordinates of the boundaries of a shape, and establish a 2D histogram at each point. This histogram is a function of the distance and the angle from each point to all other points along restricted path lying entirely inside the leaf shape. The classification accuracy reported by the authors was 85.1%. IDSC is a good classifier but the features used in this work are not enough. Large number of features should be obtained and then, the best of them should selected using certain feature selection algorithms.

Andreas et al. (2010) developed LEAFPROCESSOR- a software package which provides a semi-automatic and landmark-free method for the analysis of a range of leaf-shape parameters, combining both single metrics and PCA. Bending energy was employed as a tool for the analysis of global and local leaf perimeter deformation. The bending energy is a descriptor that provides a global measure of the curvature of the leaf perimeter and it's obtained via integration of the square of the contour's curvature along the perimeter.

Another study by Cerutti et al. (2013) proposed a specific method for the identification of compound-leaved tree species, with the aim of integrating it in an educational smartphone application. Their work was based on dedicated shape models for compound leaves and was designed to estimate the number and shape of leaflets. A deformable template approach was used to fit these models and produce a high-level interpretation of the image content. The resulting models were later used for the segmentation of leaves in both plain and natural background images, by the use of multiple region-based active contours. Combining this with other botany-inspired descriptors accounting for the morphological properties of the leaves, a classification model making use of semantic interpretation was proposed. The sample images used consist of over 1000 images from 17 European tree species. The overall methodology involves leaf color model estimation, model-based compound leaf segmentation, leaflet selection and characterization, and plant species classification.

2.6 General summary

The review of related literature on computer-based plant leaf identification systems have shown the importance of image processing and pattern recognition techniques.

Some limitations such as low discriminating power between some crop plants and weeds, rejection of variability within the same species and acceptance of variability between different species of plants, extraction of complex features such as a leaf having different colour at the back and at the front, coupled with the need for improved classification speed and accuracy, are still the major challenges facing the existing systems. Moreover, there is a need to extract more features and combine them in a single study so as to improve classification accuracy of computer-based vision system for plant species identification using the images of their leaves.

Also, of the image segmentation techniques like Sobel, Canny, LoG, Prewitt, and CNN, there is a need to use the one that offers reduced segmentation time and efficient edge extraction. Further studies are required on the application of more efficient techniques or amalgamation of techniques for increased speed and accuracy. Various features need be extracted and analysed to obtain the best of the features that will improve the accuracy of the underlying classifier. A wrapper-based feature selection algorithm has been proved to be more powerful than filtering approach and thus, will be a good choice to consider for this study. A classifier that offers great speed and accuracy will be a good choice to use. A good model that has these functionalities is a hybrid model involving CNN, PNN, and GA coupled with various image descriptors. PNN is chosen for this work due to its fast non-linear approaching ability, converging speed and insensitivity to outliers, and thus is suitable for multi-class classification applications. The proposed hybrid model in this research aims to improve on those methods reported in the literature in this chapter.

Chapter 3

Research Methodology

3.1 Introduction

This chapter discusses research methodology and phases of research to be undertaken to develop a hybrid model involving PNN, CNN, and GA for automatic identification of leaves. The chapter will first describe the proposed approach for plant recognition. This includes image acquisition, image pre-processing, image segmentation, feature extraction, and image classification. The general research methodology adopted for this study will be described in section 3.2, while the actual proposed methodlogy will be described in section 3.3 . The research phases and activities are outlined in section 3.4. This is followed by section 3.5 on details of data sets used for the study, section 3.6 computational platforms and the summary in section 3.7.

3.2 Research Methodology

A research methodology is a set of guidelines for solving a problem. The common components of a research methodology are phases, tasks, methods, techniques and tools (Basili, 1993). The form taken by a research methodology may either be experimental or analytical. As leaf classification is a quantitative domain and various databases of leaves are available for evaluation of the developed methods, an experimental methodology was proposed. The three common experimental approaches are engineering, scientific and empirical. The choice proposed for this research was the engineering method as this is one of the standard approaches in image processing applications (see Table 3.1).

Table 3.1: Engineering Approach to Research Methodology (Basili,1993)

STEP 1	STEP 2	STEP 3	STEP 4	STEP 5
Observe existing solutions	Propose Better Solutions	Develop better solution	Measure, analyze, and Evaluate	Repeat STEPS 1 to 4 until there is convergence (no further improvement)

3.3 Proposed Methodology

The proposed methodology used for this study is shown in Figure 3.1. This includes image acquisition, image preprocessing, image segmentation, feature extraction, feature selection analysis and finally, image classification. The details of these processes are described in sections 3.3.1 to 3.3.5. The methodology with the research phases of this work are also diagramatized in the flowchart shown in Figure 3.7.

3.3.1 Image acquisition

This stage involves plants leaves image acquisition using either a digital camera or scanner. The stage is used in both 'training', where selection of plant leaves are acquired to build a database for future matching, and during classification, where new leaves are acquired and matched to the images in a database. The collection of plants leaves for database construction may be done by a botanist or others who have the knowledge of plants specimen collection (Belhumeur & David, 2011), or may be sourced from an existing database.

3.3.2 Image pre-processing

This phase involves pixel-based processes such as colour-to-gray image conversion and binary image conversion of the plant leaves (see Figure 3.2). Operations on the pixel values are commonly called

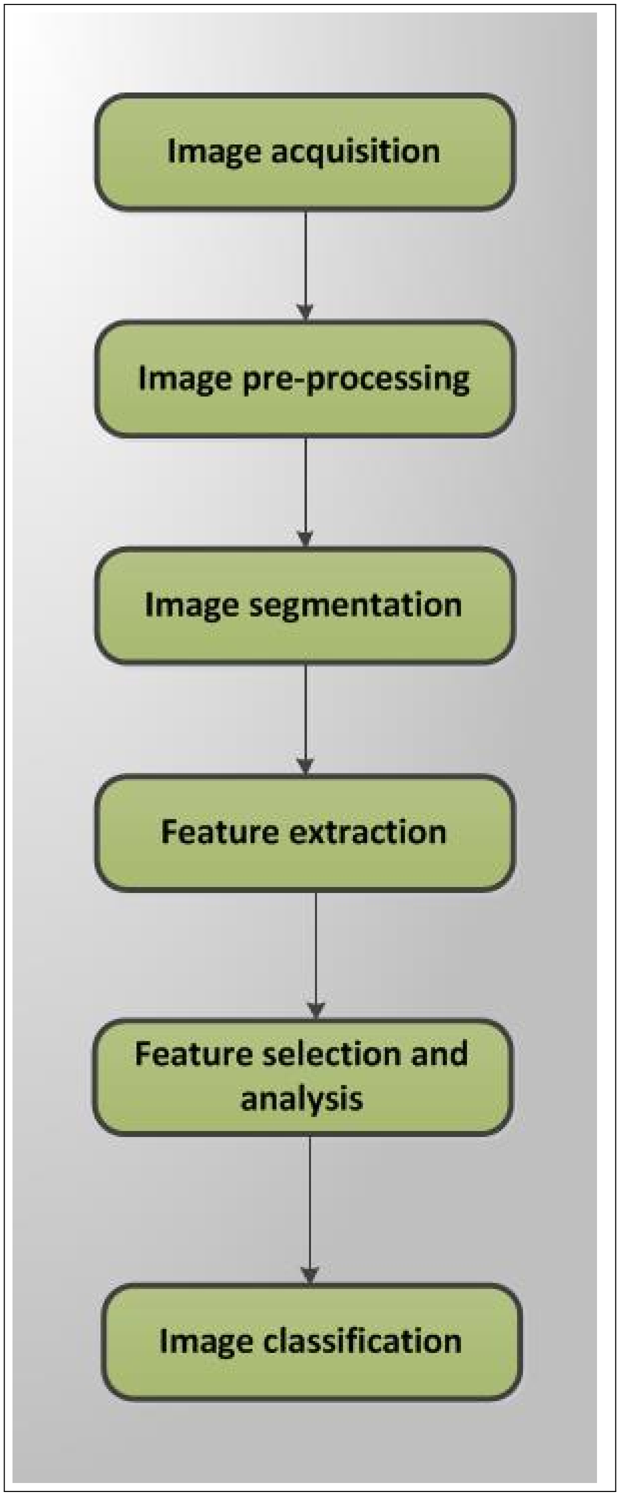


Figure 3.1: Proposed Methodology (Research Process)

gray-level reduction. This step is often required by some image segmentation techniques. As seen in section 2.5 of chapter 2, many of the existing works on computer-based plant species recognition use grayscale images of plant leaves as the input to the segmentation module (Jyotismita & Ranjan, 2011). The colour image of a plant's leaf can be converted to its gray scale equivalent image through its colour pixels value. The primary reasons why grayscale version of a digital image is often preferred for extracting some descriptors instead of using colour images directly is that grayscale images lead to algorithmic simplification and reduced computational requirements (Christopher & Garrison, 2012). The formulars in Equations 3.3.1,3.3.2, 3.3.3, and 3.3.4 were examined for image pre-processing.

1. Method 1 (Average Method): This average method simply averages the values $R+G+B$ and is given as

$$Method1 = \frac{(R+G+B)}{3} \tag{3.3.1}$$

2. Method 2

$$Method2 = \sqrt{R^2+G^2+B^2} \tag{3.3.2}$$

3. Method 3: Luminous Intensity Method: The luminosity method is a more sophisticated version of the average method. It also averages the values, but it forms a weighted average to account for human perception. Human beings are more sensitive to green than other colors, so green is weighted most heavily. The formula for luminosity is

$$Method3 = 0.299R+0.587G+0.114B \tag{3.3.3}$$

4. Method 4: Luma Method

$$Method4 = 0.210R+0.710G+0.07B \tag{3.3.4}$$

Other colometric equation for converting colour image to grayscale are found in the appendix (Equations 9.4.38 to 9.4.43).

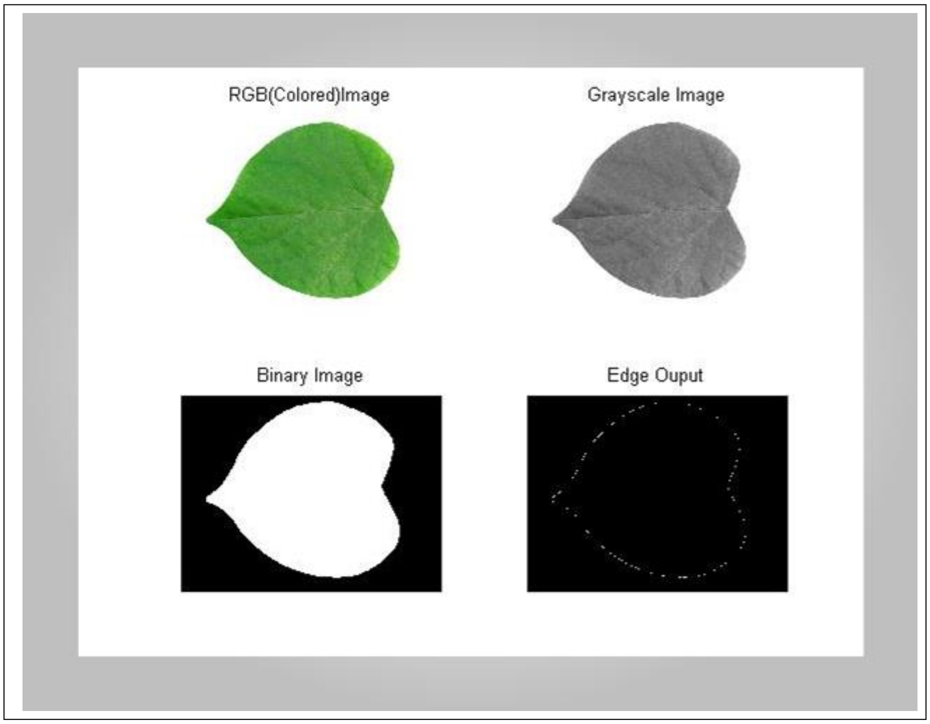


Figure 3.2: Image pre-processing and segmentation

3.3.3 Image segmentation

This step follows image pre-processing and also precedes feature extraction. Image segmentation is the process of splitting a digital image into a number of regions (subsets). This involves looking for compatible pixel class. From a mathematical point of view, the segmentation of an image, say, image $f(x,y)$ of a plant's leaf, is a partition of $f(x,y)$ by a partition operator Θ_{PART} into smaller images $\{f_i\}^n{}_i = 1$ such that the following are satisfied.

1. $$\Theta_{PART} f(x,y) = \biguplus f_i = f(x,y) \tag{3.3.5}$$

2. $$f_i \cap f_j = \phi, i \neq j \tag{3.3.6}$$

3. Each $f_i, i = 1(1)n$ satisfies a predicate or set of predicates.

A binarized version of each images are often used by many image descriptors since it's easier for the descriptors to work with. CNN (edge detection templates) and other conventional edge operator such as Canny, Prewitt, LoG, and Sobel were employed in this study and used in conjuction with Fourier Descriptors.

3.3.3.1 Image segmentation using conventional edge operators

The goal of image segmentation is to find regions of interest (ROI) in the images. The division of images into ROIs is often necessary before any processing can be done at a higher level than that of the pixel (Umbaugh, 2011).

The gradient edge detector

For any 2D image $f(x,y)$ its gradient operator is defined as the vector of first-order partial derivatives given as:

$$\nabla f(x,y) = \left[\frac{\partial f}{\partial x}(x,y) \quad \frac{\partial f}{\partial y}(x,y)\right] \tag{3.3.7}$$

while its gradient magnitude and direction are respectively given as:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{3.3.8}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{M_y}{M_x}\right) \tag{3.3.9}$$

where

$$M_y = \frac{\partial f}{\partial y} \tag{3.3.10}$$

and

$$M_x = \frac{\partial f}{\partial x} \tag{3.3.11}$$

Using central difference scheme from numerical method, the 2nd derivative (x-component and y-component) for the images used are given as :

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \tag{3.3.12}$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \tag{3.3.13}$$

The combination of the x-component and y-component 2nd derivatives gives

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{3.3.14}$$

If other numerical descritization schemes are used , then the following matrices are generated:

$$\mathrm{M1} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \mathrm{M2} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \mathrm{M3} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & +4 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \mathrm{M4} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

CNN Templates:

$$\mathrm{A} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathrm{B} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{pmatrix}, \boldsymbol{I} = (-1) \tag{3.3.15}$$

Sobel Operator:

$$\mathrm{H_x} = \begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}, \mathrm{H_y} = \begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{3.3.16}$$

Prewitt operator:

$$\mathrm{H_x} = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{pmatrix}, \mathrm{H_y} = \begin{pmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{pmatrix} \tag{3.3.17}$$

Laplacian operator:

$$\mathrm{L_1} = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{pmatrix}, \mathrm{L_2} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{3.3.18}$$

The LoG is derived by convoluting a given image $f(x,y)$ with by a Gaussian kernel in Equation 3.3.19

$$g(x,y,\sigma) = \frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.3.19}$$

at a certain scale σ to give a scale space representation

$$L(x,y;\sigma) \ = g(x,y,\sigma) * f(x,y) \tag{3.3.20}$$

The Laplacian operator is expressible as Equation 3.3.21:

$$\nabla^2 L = L_{xx} + L_{yy} \tag{3.3.21}$$

Canny operator

The Canny operator involves the following steps

1. application of Gaussian smoothing
2. application of Sobel operator
3. application of nonmaximal suppression
4. thresholding with hysteresis to connect the edge pixels.

A threshold (see Figure 3.3) can be applied to some of these operator so that only a fraction of the gradients present in the image are retained. Herein, a binary image which compose of $1's$ and $0's$ are then retained. The binary image display all the points in the image where its gradient is larger than the value of the threshold that was imposed on the operation.

$$I(i,j) = \begin{cases} 1 \quad \text{if} \quad |Grad(i,j)| > h \\ 0 \quad \text{if} \quad |Grad(i,j)| < h \end{cases} \tag{3.3.22}$$

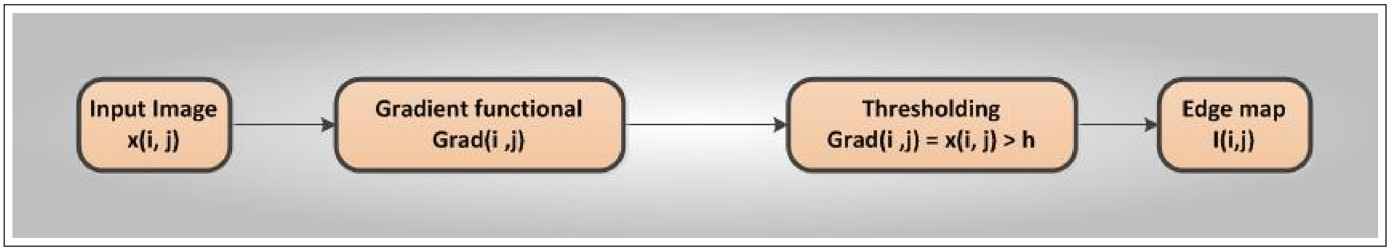


Figure 3.3: Binarization using edge functionals

A pixel p at coordinates (x,y) has four horizontal and vertical neighbors whose coordinates are given by $(x+1,y),(x-1,y),(x,y+1),and\ (x,y-1)$. This set of pixels are called 4-neighbors. The four diagonal neighbors of p are $(x+1,y+1),(x+1,y-1),(x-1,y+1),and(x-1,y-1)$. The CNN templates used for edge detection was a 8-neighbors based pixels template.

3.3.3.2 Image Segmentation Using Genetic CNN

Overview of Cellular Neural Networks

Cellular Neural Networks (CNN) are variants of Artificial Neural Networks (ANN) with the distinguishing feature of only neighborhood communication (Chua & Yang, 1988). CNN is an electrical circuit shown in Figure 3.4) and was invented in 1988 by Chua and his graduate student Yang at the Department of Electrical Engineering and Computer Sciences, university of California, Berkeley (Chua & Roska, 2002). A standard CNN topological structure is made up of an $M \times N$ or rectangular array of cells $C(i,j)$ (or dynamic components) with Cartesian coordinates (i,j), $i = 1(1)M, j = 1(1)N$ as shown in Figure 3.6. CNN is a hybrid model, sharing features from both Cellular Automata (CA) and ANN (Hezekiah et al., 2010). The circuit structure and element values of all cells of a CNN are homogenous while the Equation 3.3.23 governing the behavior of a CNN cell circuit is a dynamical system (or Ordinary Differential Equation (ODE)) derived from evolution laws and circuit theory as shown in Figure 3.4 and Figure 3.5. The output of the circuit (also called the amplifier), is given in Equation 3.3.24

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{(k,l)\in\mathcal{N}(i,j)} A(i,j;k,l)\cdot y_{kl}(t) + \sum_{(k,l)\in\mathcal{N}(i,j)} B(i,j;k,l)\cdot u_{kl} + I(i,j;k,l) \qquad (3.3.23)$$

$$y_{ij}(t) = f(x_{ij}(t)) = \frac{1}{2}(\left|x_{ij}(t)+1\right| - \left|x_{ij}(t)-1\right|) \tag{3.3.24}$$

The variables appearing in Equations 3.3.23 and 3.3.24 are defined in Table 3.2, while a visual diagram of a CNN circuit is shown in Figure 3.4

Table 3.2: Definitions of variables in CNN circuit equation (Hezekiah et al., 2010)

S/N	Variable	Definition
1	I	Independent voltage source or the network bias term
2	C	Linear Capacitor
3	R_x	Linear Resistor
4	x_{ij}	Internal state of a cell (i,j)
5	u_{kl}	Input voltage to a cell (k,l)
6	$A(i,j;k,l)$	Feedback template (weighted sum of output voltages of all neighborhood cells)
7	$B(i,j;k,l)$	Feedforward template(Linear voltage-controlled current sources from neighborhood cells)
8	y_{kl}	The output or amplifier of a cell (CNN circuit)
9	$N_r(i,j)$	Symbol for neighbourhood

It is noteworthy, by the reason of voltages measured across the circuit (max voltage = +1, min voltage = -1), that the constraints $|x_{ij}(0)| \leq 1,\ 1 \leq i \leq M; 1 \leq j \leq N$, and $|u_{ij}(0)| \leq 1, 1 \leq i \leq M; 1 \leq j \leq N$ are fundamentally imposed on the internal state of neuron and input of neuron, where M and N are the horizonal and vertical dimension of the CNN grid. According to Roska, Zarandy, and Rekeczky (2003) , there is a one-to-one mapping between the pixels of images and the CNN cells. Black is coded as +1V, white as -1V and grayscale as $g = \{g \in R : -1 \leq g \leq +1\}$. The several topological layouts for a CNN grid are rectangular, triangular, or hexagonal geometry, a $2D$ or $3D$ torus, a $3D$ finite array or a $3D$ sequence of $2D$ arrays (layers) but 2D CNNs organized in an 8-neighborhood rectangular grid remains the most popular. The state of each cell and its output are functionally determined by the input and the output of its neighbour cells, and the initial state of the network. The parameters of a CNN, conventionally denoted as $\{A,B,I\}$ are called the cloning templates, where A, B, and I are already defined in Table 3.2. For instance, a 5-by-5 neighbourhood CNN contains $((5\times5)+(5\times5)+1 = 51)$ elements.

With different entries in $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{I}\}$, a CNN can bring about a large number of complex dynamical behaviour, as stated by Ercsey, Maria, Neda, and Roska (2008). In other words, the layout of the elements in the templates $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{I}\}$ are called boundary conditions, and thus, we may say that the correct behavior of any CNN model is majorly a factor of the imposed boundary conditions (Tamas and Giovanni, 2009). According to Aizenberg (2001) three good examples of CNN templates that implement mean filter and the two spatial part of the well known Sobel edge (see Matrices 3.3.16) detection operator respectively are:

$$\mathrm{A} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathrm{B} = \begin{pmatrix} 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \end{pmatrix}, \boldsymbol{I} = (0) \tag{3.3.25}$$

and

$$\mathrm{A}^1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathrm{B}^1 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{pmatrix}, \boldsymbol{I}^1 = (0) \tag{3.3.26}$$

$$\mathrm{A}^2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathrm{B}^2 = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \boldsymbol{I}^2 = (0) \tag{3.3.27}$$

There are a number of other different templates that implement other useful image processing techniques as seen in (Chua & Roska, 2002). Any image processing CNN, most often than not, go back to a constant steady state after any transient state that has already been initialized by a given input image and generally, and if using Runge-Kutta method, the last state of one cell(a dynamical system) can be represented by Equation 3.3.28 (H. O. Babatunde et al., 2012).

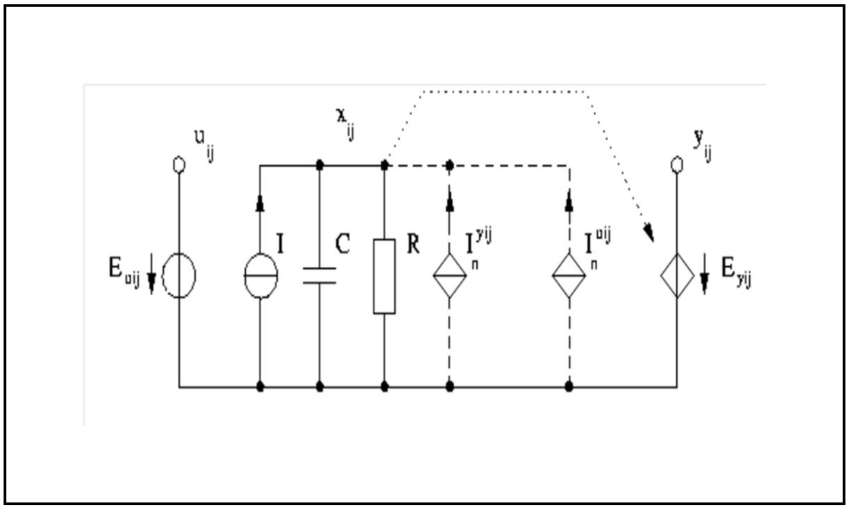


Figure 3.4: Circuit representing Cellular Neural Networks (Hezekiah et al., 2010)

$$x(t) = x(t_0) + \int_{t_0}^{t} \dot{x}(\tau)d\tau = x(t_0) + \int_{t_0}^{t} f(x(\tau))d\tau \tag{3.3.28}$$

where

$$\int_{t_n}^{t_{n+1}} f(x(t))dt = \frac{k_1 + 2k_2 + 3k_3 + k_4}{6} \tag{3.3.29}$$

and

$$k_1 = \Delta t . f(x(t_n)) \tag{3.3.30}$$

$$k_2 = \Delta t . f(x(t_n) + \frac{1}{2}k_1) \tag{3.3.31}$$

$$k_3 = \Delta t . f(x(t_n) + \frac{1}{2}k_2) \tag{3.3.32}$$

$$k_4 = \Delta t . f(x(t_n) + \frac{1}{2}k_3) \tag{3.3.33}$$

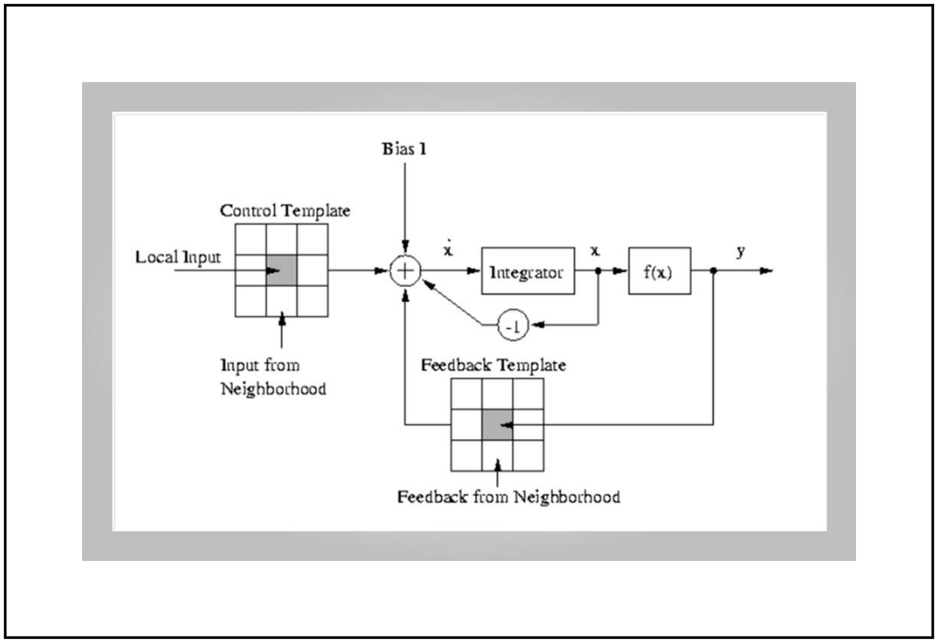


Figure 3.5: The Block Diagram of a CNN Cell (Hezekiah et al., 2010)

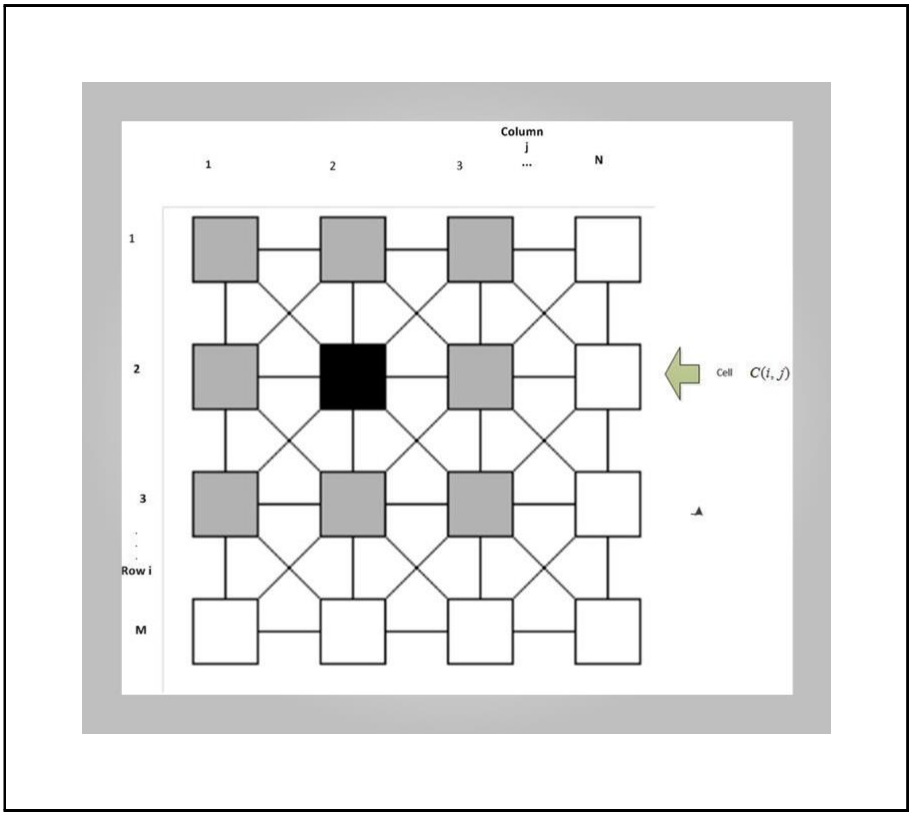


Figure 3.6: A CNN Rectangular Grid with M rows and N columns (Hezekiah et al., 2010)

3.3.4 Feature extraction

Any digital image has associated features like shape, color, size, orientation and texture. In the same way, a plant's leaf image has associated features as shown in Table 2.1. Features such as color moments, moments invariants images, geometric and morphological features are extracted at the feature extraction stage and this is the stage that precedes the classification stage. In other words, the outputs of the features extraction module are used as input to the classifier either for training or for classification.

3.3.5 Image Classification

This is the last stage in computer-based plant recognition systems. The output from the features extraction modules are used as input to train classifiers to classify plants species. The classification model in this study was based on genetically optimized PNN which was trained with GA-based features.

3.4 Research Approach

This research was modularized into three phases viz phase 1 to 3, each containing a different number of tasks, while the tasks were named according to the phases as shown in Figure 3.7. In other words, Task 1.1, Task 1.2, and Task 1.3 belong to phase 1, Task 2.1 to phase 2, while Task 3.1, 3.2 and 3.3 belong to phase 3. At the end of phase 3, performance of the CNN-PNN-GA model was evaluated.

3.4.1 Research Phase 1

This is the first phase of the research process in this study. The tasks performed in this phase were image pre-processing, image segmentation, and feature extraction. The source of images of leaves that were used in this study were taken from the Flavia dataset which is publicly available (Wu et al., 2007).

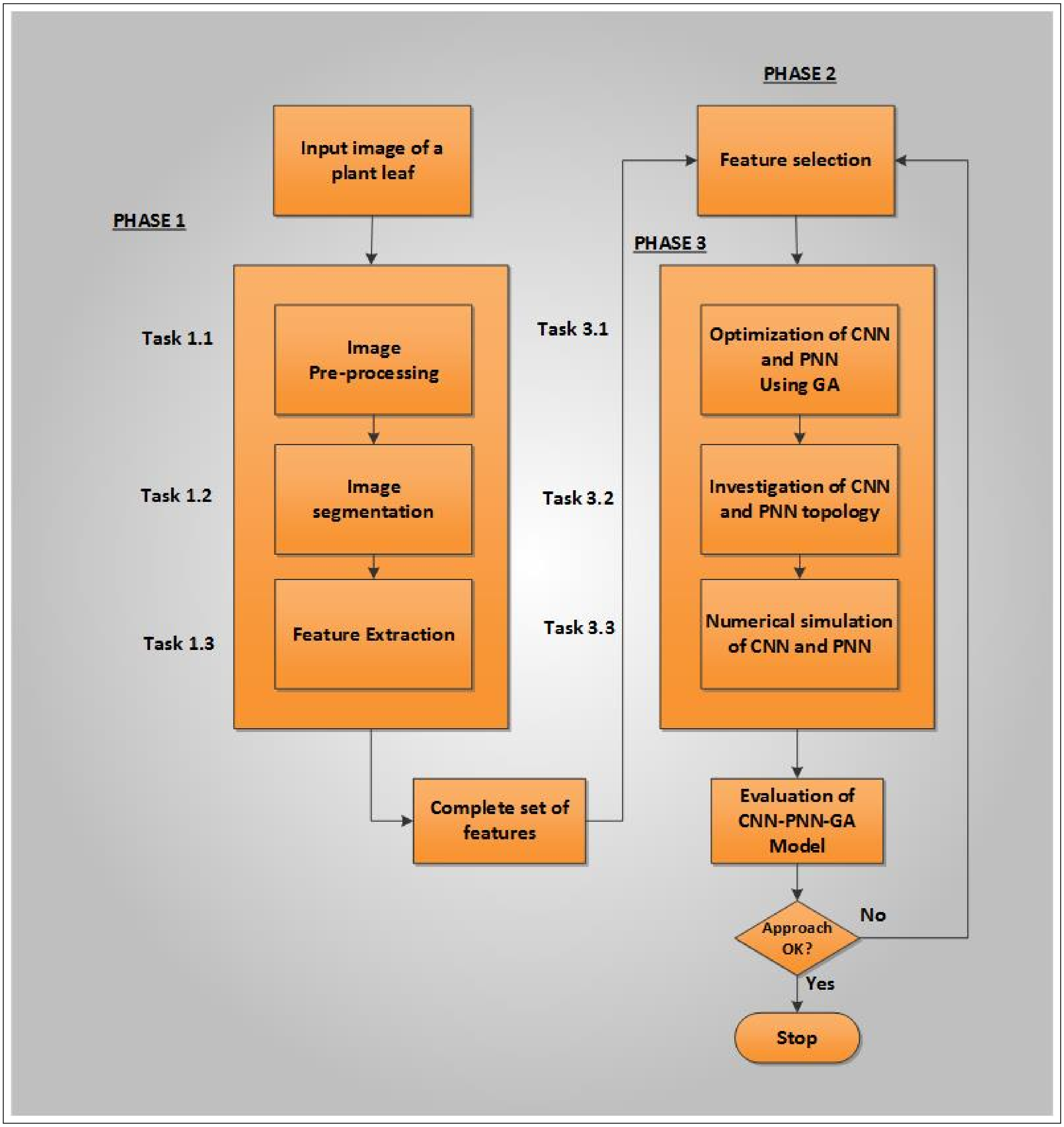


Figure 3.7: A Flowchart showing the phases of the proposed CNN-PNN-GA framework

3.4.1.1 Task 1.1 (Image pre-processing)

This task involved preparation of the images to improve the effectiveness of the subsequent segmentation task. The colour pixel components of the image can be manipulated to get a gray-scale image. This step simplifies subsequent computations such as shape descriptors and segmentation, that do not rely on colour. Examples of such conversion were discussed in Section 3.3.2 of Chapter 3. An investigation was carried out as to which conversion formula to use in order to support a more robust segmentation. It should be noted that, even though grayscale images may be used, this does not preclude subsequent feature extraction from the original colour image, masked by the segmented region. Some other tasks that were also performed at this stage are image resizing and noise removal (see Figure 3.8)

3.4.1.2 Task 1.2 (Image segmentation)

Image segmentation was discussed in Section 3.3.3. This involves partitioning a digital image into a number of segments so as to transform pre-processed image into distinct component from which features can be extracted (see Figures 3.2 & 3.8). The resulting image segments here are also termed Region Of Interest (ROI). The tasks here include differentiating between leaf and background and also between individual component of the leaf. The same procedures were applied on both training and test data set. An investigation was carried out to select the appropriate techniques that will produce robust segmentation of leaf images in this study. CNN-based segmentation method was also benchmarked against other segmentation techniques such Canny, Sobel, LoG and Prewitt.

3.4.1.3 Task 1.3 (Feature extraction)

Once the image of a leaf was segmented, features were extracted from the segmented regions or ROIs. Feature extraction is basically a mapping from an m-dimensional input space to n-dimensional feature space, given by $\Phi : R^m \rightarrow R^n, \ (n \leq m)$. This mapping may be linear or non-linear and can be acquired through various methods depending on the feature type. The

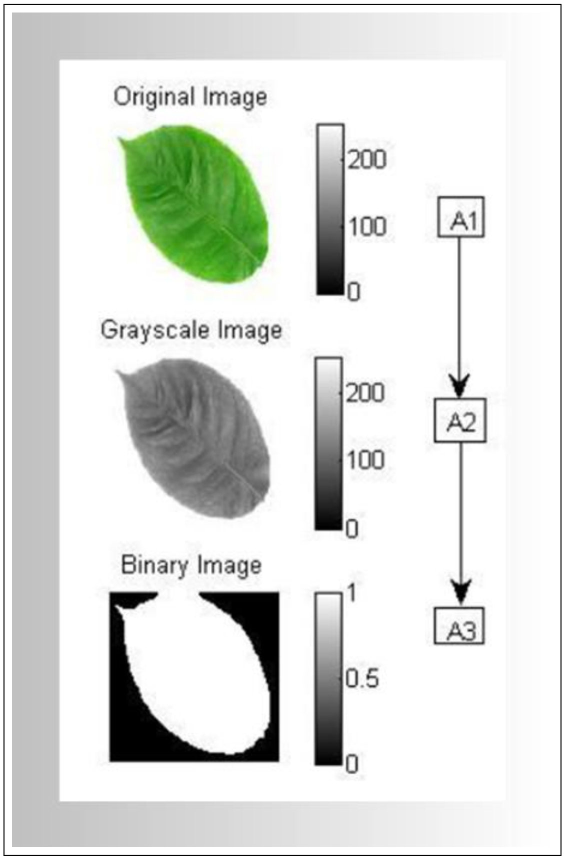


Figure 3.8: Image preprocessing and segmentation

set of features from this stage can then be utilised by the subsequent classifier for training and classification. The features used in this research are outlined and described in Chapter 4. Some of the extracted features may turn out not to be informative and thus, not all features were used in the classification phase. Some desirable characteristics of good feature descriptors are (1) uniqueness (representing the object uniquely), (2) completeness (unambigous representation) (3) invariance with respect to translation, rotation and scaling (TRS) (4) robustness to noise. A commonly used technique which may be used to test for class-discriminating capability of a specific feature is known as the receiver operating characteristic (ROC). The ROC is an overlap between the pdf of the feature data in the training and test data set (Sergios & Koutroumbas, 2010). The program module associated with this section was called ”features provider”.

3.4.2 Research Phase 2

This phase comprises several tasks for addressing sub-research questions 1, 2 and 3. Once tasks 1.1 to 1.3 in the Phase 1 of the research were completed, task 2.1 of phase 2 commenced. The task in this phase addresses sub-research question 1.

3.4.2.1 Task 2.1 (Investigation of features selection techniques)

This task was about investigation of (and selection from) the set of features produced in Task 1.3. Out of the features generated from Task 1.3, some were not very informative or were strongly and mutually correlated. This means that some of the features might not be useful for generating the classifier. Different combinatorial set of features were obtained in this phase in order to keep the best combination. Ideally, the dimension of the features set should be relatively small with respect to the dimension of the training set and so, appropriate tools (subspace or manifold projection techniques) such as PCA, Multi-Objective Evolutionary Algorithm (MOEA) as seen in Bruzzone and Persello (2010) and Particle Swam Optimization (PSO) will be explored for finding the suitable features or combinations of set of features for classification amidst different factors such as leaf maturity, deformity, image translation, rotation and scaling. The output of the feature selection modules in this project was a number of features (which is dependent upon the manifold projection techniques) with maximum classification accuracy.

3.4.3 Research Phase 3

The input to this phase was the set of features from Task 2.1 and it is envisaged that knowledge developed in this phase will be one of the main contributions for this project. The tasks in this phase were GA-Based Optimization of parameters for CNN and PNN, investigation of CNN and PNN topology, and numerical solution of CNN and PNN. Tasks 3.1, 3.2 and 3.3 in this phase were related and involved a number of iterations which may not be necessarily sequential in execution with respect to each other.

3.4.3.1 Task 3.1 (GA-Based Optimization for CNN and PNN)

This task was iterative and involved application of GA to obtain the parameters of CNN and PNN needed to obtain optimal segmentation and classification results respectively. Investigation was carried out to find the most suitable classification model using the features obtained from Task 2.1. However, any classifier which was chosen contained the knowledge of each pattern category (image descriptors and class information associated with each plant species) and also, the discriminating metric among all patterns (leaf types or plant species).

(a) Important questions that were considered included:

1. What are the parameters of the CNN & PNN to be optimized by the GA?
2. What are the appropriate GA-encoding type best suitable for the optimization of CNN & PNN parameters respectively?
3. How do we construct an appropriate fitness or objective function for the CNN-GA, PNN-GA and respectively?
4. What are the appropriate GA parameters (i.e selection mechanism, genetic operators and termination condition) for the optimization tasks associated with CNN and PNN respectively?

After the completion of this tasks, results obtained were used in the development of the classification model.

3.4.3.2 Task 3.2 (CNN and PNN Topology)

(a) With regards to the application of CNN in this study, the following questions were addressed:
(1) The dimension of training parameters (CNN templates) to be used?
(2) The type of activation function to use?
(3) The CNN type (Discrete Time CNN (DT-CNN) or Continuous Time-CNN (CT-CNN)?

An investigation was carried out to determine template size to use for our CNN (3-by-3, 5-by-5, etc), the activation function, the type of the CNN (DT-CNN or CT-CNN), numerical methods to be employed to discretize the CNN. These factors would greatly influence the accuracy of the proposed hybrid system.

(b) In considering the PNN-based classifier in this study, the following issues were addressed:
(1) The type of radial basis function (variants of Gaussian) in the activation function?
(2) The size of Parzen window (PNN smoothing parameter).
(3) Number of nodes in the pattern layer?

3.4.3.3 Task 3.3 (Numerical Schemes for CNN and PNN)

This section involved an investigation into some different numerical methods where appropriate discretization scheme were sought to solve the CNN and PNN. The factors considered includes
(1) Numerical scheme for computing the decision region between the test and training dataset
(2) The type of numerical scheme to use to solve the CNN

3.4.4 Evaluation of the proposed Classifiers obtained from Phase 3

The suitable set of features from Task 2.1 were used with CNN and PNN to form the classifiers in an iterative fashion. The three metrics used to evaluate the performance of the classifiers in the proposed Neuro-Genetic Hybrid Intelligent System (NGHIS) were:

1. Accuracy

   The performance of the classifiers were evaluated against the test data set using the error rates (computed using confusion matrix). The number of wrong and correct classification were measured.

2. Robustness

   The classifiers were evaluated against factors such as translation, rotation, scaling of leaf images, plant species with complex features and deformed leaves.

3. Speed

   Computational time required by the classification model.

3.5 Data

This section summarises the databases to be used. This work was primarily based on the Flavia dataset and another image dataset comprising of only binary images.

3.5.1 First dataset (The Flavia dataset)

The Flavia dataset is composed of a set of highly constrained leaf images taken against a white background and without any stem. This 1GB dataset contains 1907 images of 32 species of plants (Wu et al., 2007) which are available at (http://flavia.sourceforge.net/) and each of which has a file name represented by a 4-digit number, followed by a ".jpg" extension. Each class of leaves (with red, green and blue channels) has 50 to 77 leaf samples having resolution of 1600×1200 pixels (see Figures 3.10 &3.11).

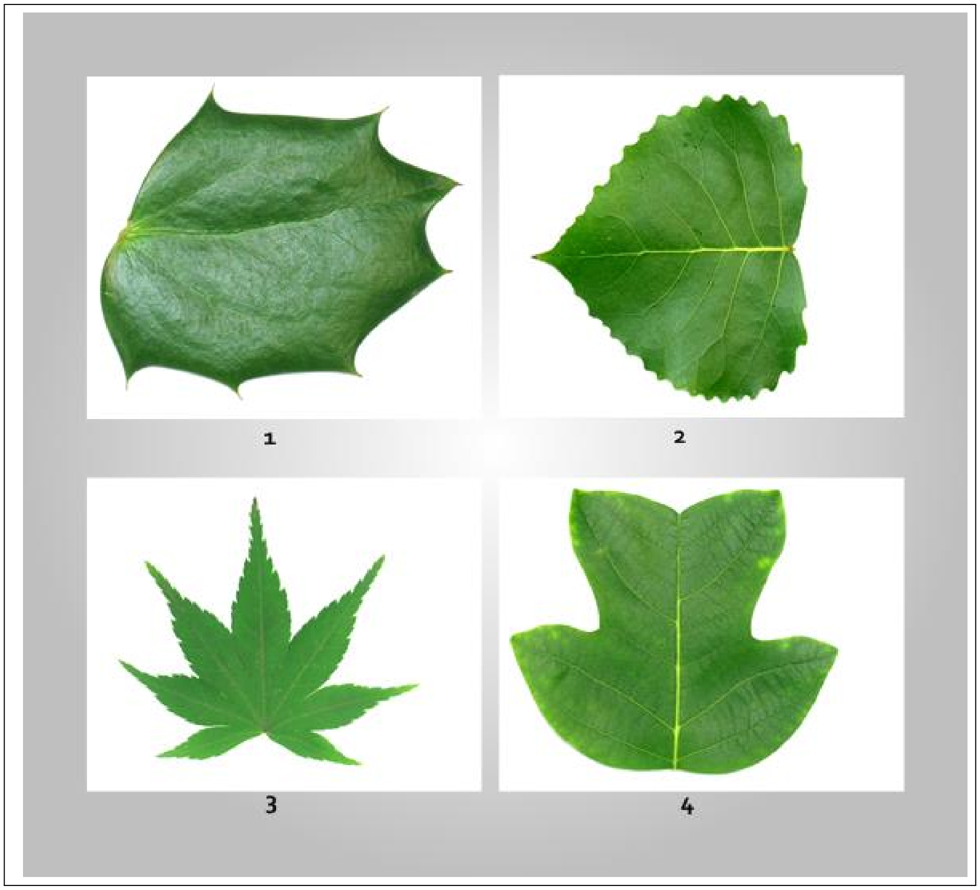


Figure 3.9: Four selected samples from the Flavia dataset

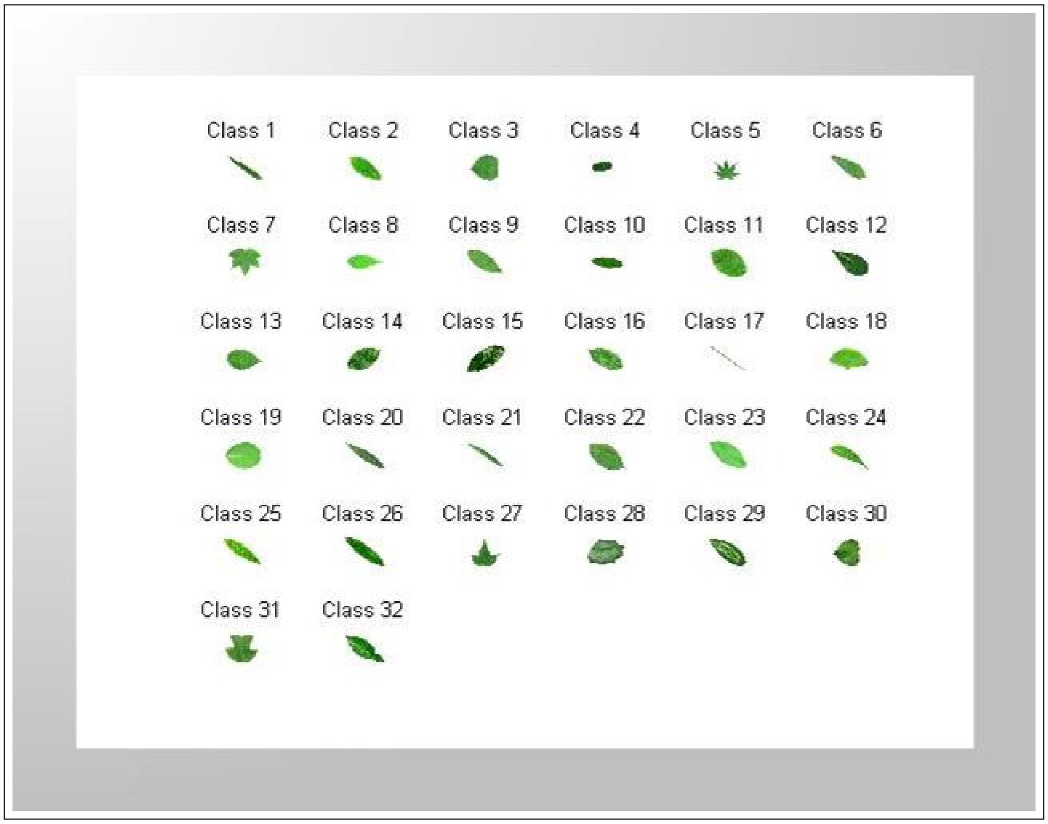


Figure 3.10: Classes of Leaves in the Flavia dataset

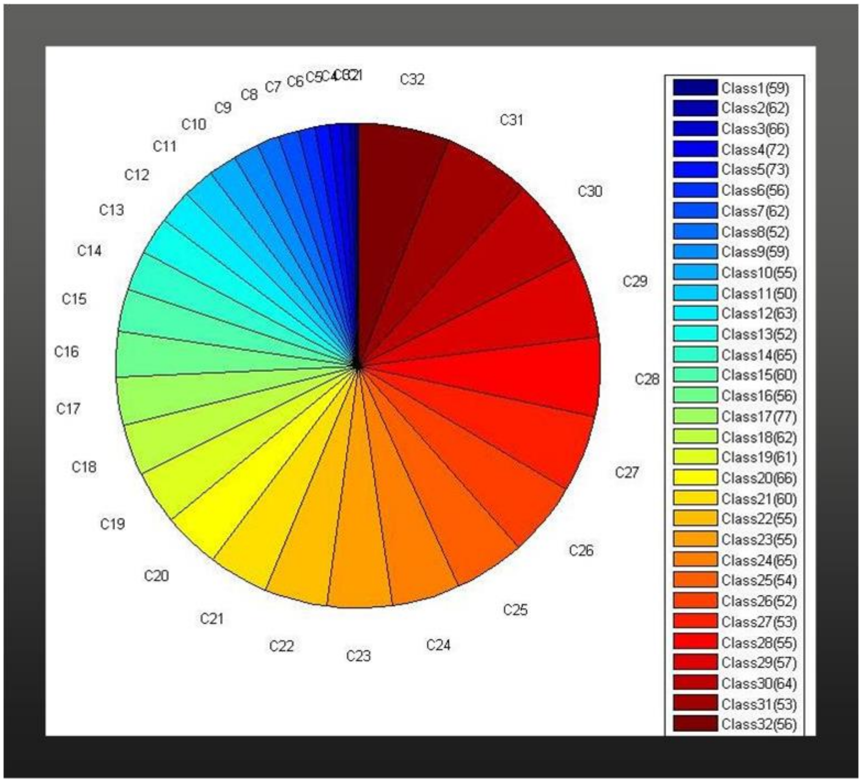


Figure 3.11: Proportion of plant species in the Flavia dataset

3.5.2 Second dataset

The second data set (Orwell, Mallah, & Cope, 2012) comprised leaves from one-hundred species of plants. For each species, there are sixteen samples (specimens), originally photographed as a colour image on a white background. Figures 3.13 shows respectively the entire range of species with four classes of images displayed for readabillity. The names of these species are shown in the Table 7.3 of Chapter 7.

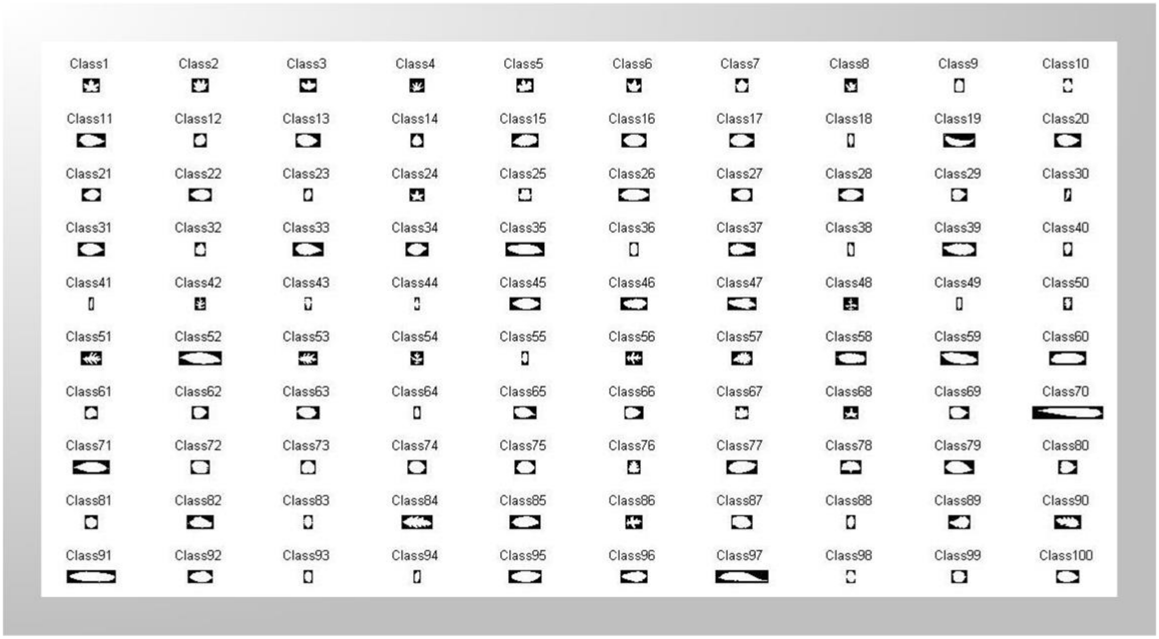


Figure 3.12: Classes of leaves in the 2nd dataset

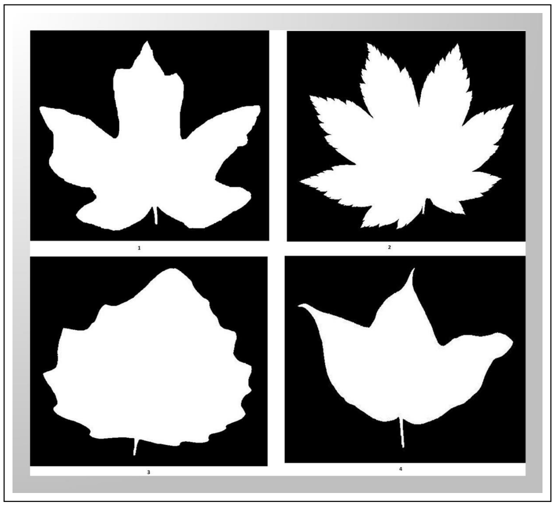


Figure 3.13: Four leaf samples from the 2nd dataset

3.6 Computational Platforms

The choice of programming language was MATLAB (MATrix LABoratory). MATLAB was developed by Mathworks (2015) and is a multi-paradigm numerical computing environment (platform) that is very flexible and adaptable for visualization, mathematical algorithms and contains lots of functions for matrix and image manipulation, and other scientific computation. It is considered to be good for creation of user interfaces, and interfacing with programs written in other languages, including Python, Java, C, C++ and Fortran.

3.7 Summary

The proposed study aimed to investigate the application of a Neuro-Genetic Hybrid model to classify plants species using the images of their leaves. Most importantly, the four main issues that characterises the methodology in this study are morphometric methods (features representation), image processing methods (image analysis and feature extraction), feature selection and analysis, and computational methods (solution of computation models (the concerned learning machines)). The basic steps required were image acquisition, image pre-processing, image segmentation, feature extraction, feature selection and analysis and image classification using the CNN-PNN-GA classification model. The CNN was used for image segmentation while the PNN was used as the classifier. A GA was used to select the best set of features from the original dataset in order to improve classification accuracy of the concerned classifier. The parameters of CNN and PNN were optimized by GA for performance improvement. To further analyse the performance the GA, PSO and PCA were also used for feature selection. The performances of the three feature reduction techniques were later compared.

Chapter 4

Image Pre-Processing, Segmentation and Feature Extraction

4.1 Introduction

This chapter describes image pre-processing, segmentation and descriptors used as well as the numerical results of such descriptors. This chapter is the key feature for the success of the image classification system as all quantified metrics derived from the steps or stages of this chapter are eventually and unavoidably used by the classifier. Image pre-processing is covered in section 4.1.1. Section 4.1.2 covers image segmentation techniques such as CNN, Sobel, Canny, Prewitt and LoG while feature extractions are covered in section 4.1.3. The features extracted were categorised into shape, colour and texture (see Figure 4.8). Findings from this chapter have been published in (O. Babatunde, Armstrong, Leng, & Diepeveen, 2014d, 2014a; O. Babatunde, Armstrong, Diepeveen, & Leng, 2015).

4.1.1 Image pre-processing

The formulas in Equations 3.3.1, 3.3.2, 3.3.3, and 3.3.4 of section 3.3.2 from chapter 3 were examined for image pre-processing, and investigation revealed that the formula shown in Equation 3.3.3 was best in that best gray scale image was derived from this formula. The image resolution derived using this preprocessing method was better than other preprocessing methods since it gave the best edge output.

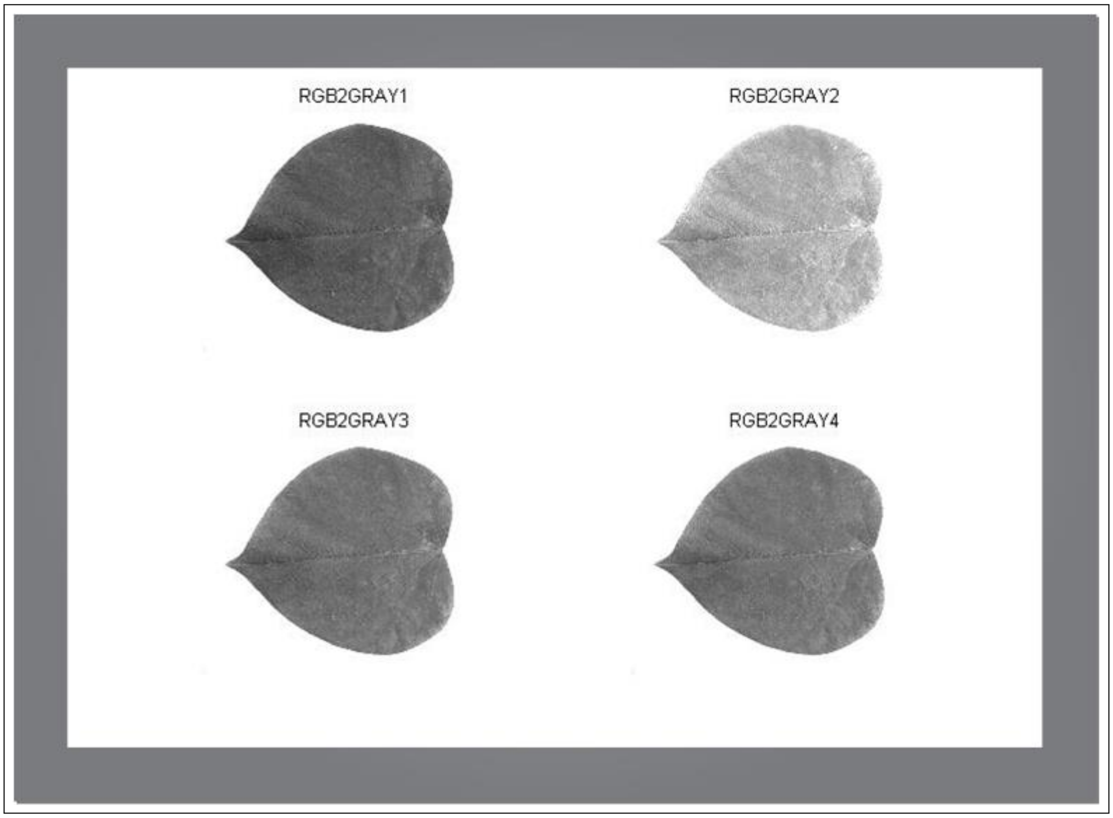


Figure 4.1: Comparison of different rgb-to-gray methods

4.1.2 Image segmentation

The images were segmented in several ways. There was gray to binary conversion and binary to edge conversion (see Figure 4.2). The image features that required the use of edge pixels were the FDs. So the results of common edge of operators was compared with the GA-based CNN edge operators in terms of quality edge pixels and computation time. GA-based CNN edge operators outperformed the conventional edge operators. The Figures 4.5 & 4.6 show the edge output based ordinary and genetic CNN templates respectively. Both of the edges shown in Figures 4.5 & 4.6 were derived faster than using the conventional edge operators shown in Figure 4.3.

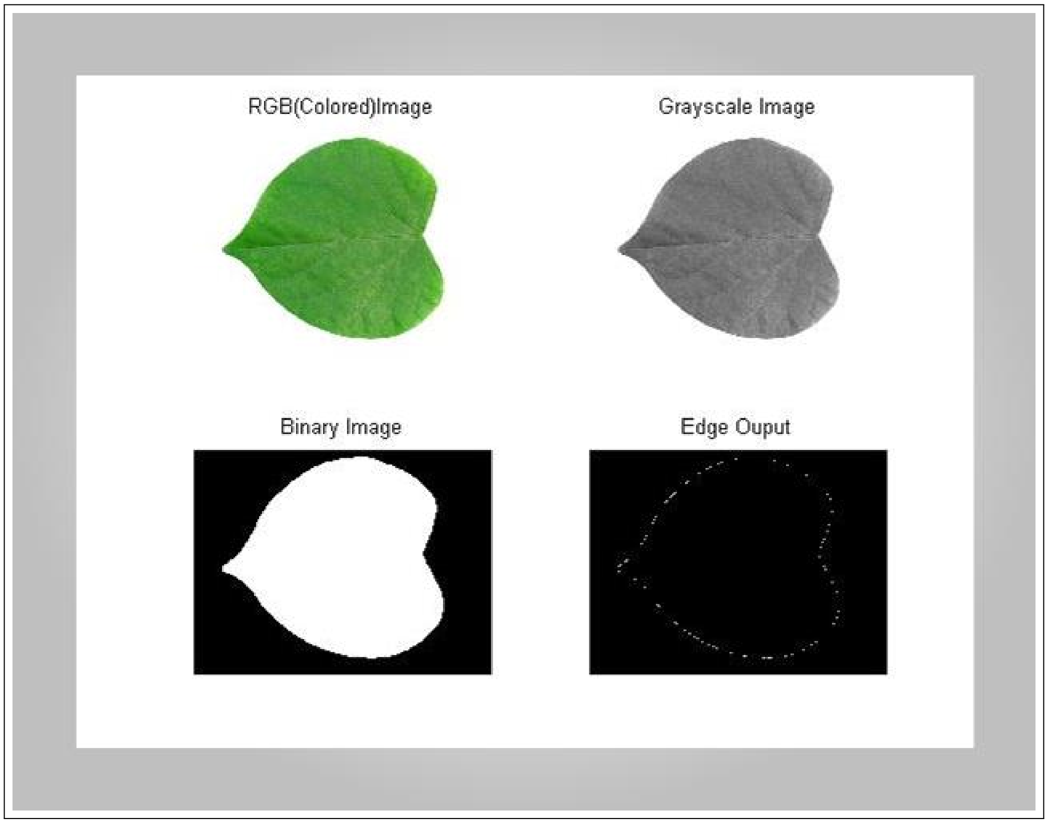


Figure 4.2: Image pre-processing and segmentation

4.1.2.1 Genetic Optimization of CNN Templates

The CNN templates are the matrix coefficients for the systems of equations in 3.3.23 & 3.3.24. Runge-Kutta (R-K, see Equations 3.3.28, 3.3.29, 3.3.30, 3.3.31, 3.3.32, & 3.3.33) method was used in discretizing the ODE in Equation 3.3.23. A sample edge output using the CNN templates shown in Equation 3.3.15 is shown in Figure 4.5. The template design process for the CNN is shown in Figure 4.4 using the GA parameters in Table 4.1. The fitness function used by the GA-CNN process is given in Equation 4.1.1 where I_{ij} and O_{ij} are the evaluated and input image pixels respectively. The best region of interest was obtained at generation 101 as shown in Figure 4.7.

$$f(CNN) = 1 - \left[\frac{\sum_{i=1}^{N} \sum_{j=1}^{M} (I_{ij} - O_{ij})^2}{8NM} \right] \tag{4.1.1}$$

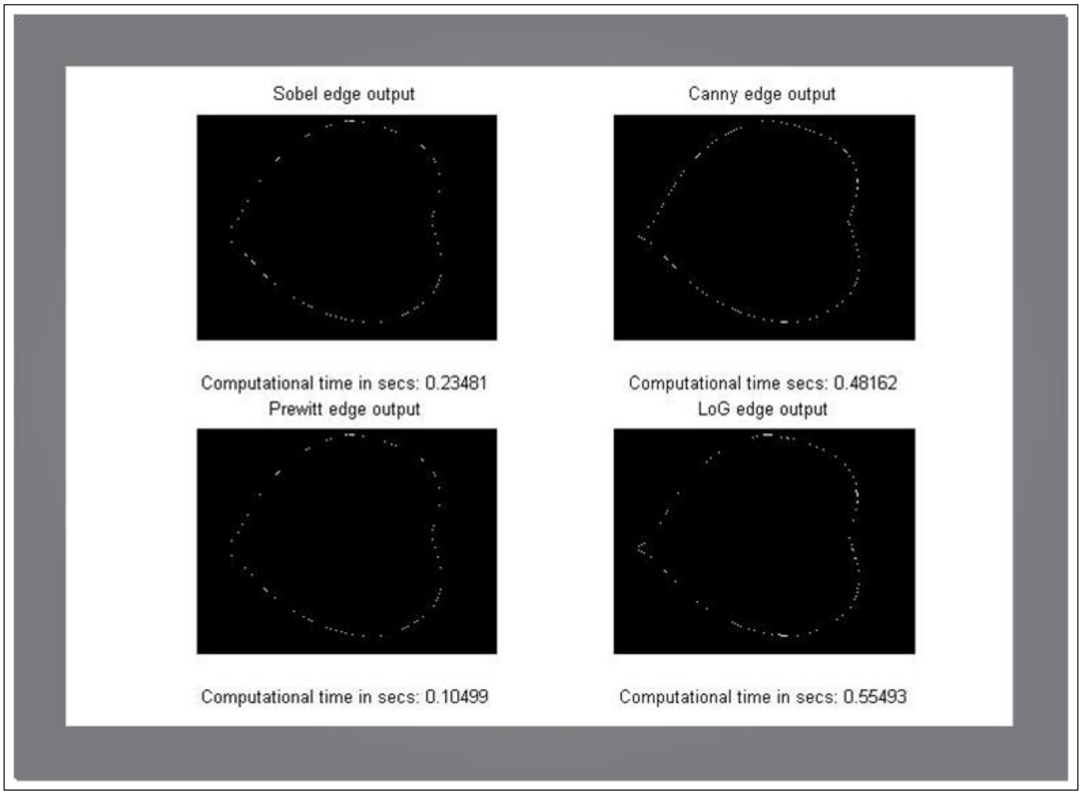

Sobel edge output
Canny edge output
Computational time in secs: 0.23481
Computational time secs: 0.48162
Prewitt edge output
LoG edge output
Computational time in secs: 0.10499
Computational time in secs: 0.55493

Figure 4.3: Edge outputs from conventional edge operators

CNN Templates (Genetically optimized):

$$\mathrm{A} = \begin{pmatrix} -0.0188 & -7.2196 & -1.6024 \\ -2.2306 & 20.8999 & -2.2306 \\ -1.6024 & -7.2196 & -0.0188 \end{pmatrix}, \mathrm{B} = \begin{pmatrix} -0.0397 & 0.3402 & -0.0362 \\ -0.2233 & -0.2497 & -0.2233 \\ -0.0362 & 0.3402 & -0.0397 \end{pmatrix}, I = (-3.3014)$$

(4.1.2)

The main idea behind the CNN was to input an image and discretize the ODE using appropriate numerical method. The R-K method was used to solve the ODE during both GA simulation and CNN image segmentation. The final templates shown Equation 4.1.2 were then used for image segmentation. As documented by Alireza, Jean, and Kyandoghere (2011) and Duraisamy and Duraisamy (2012), the relationship between the input image and the segmented image in the CNN is expressed as Equation 4.1.3.

$$dx = -x + conv(x,A) + conv(x,B) + I \tag{4.1.3}$$

where dx is the edge/segmented image (or ROI), x is the original image. The function *conv* is a convolution operator. The convolution process requires overlaying the templates masks on the input image, multiplying the coincident values and summing over all the results. This is equivalent to finding the vector inner product of the mask with the underlying subimage. The elements (or matrices) of the triple $\{A(i,j;k,l), B(i,j;k,l), I_{ij}\}$ are the cloning templates for the CNN Model or ODE in Equation 3.3.23. These three (3) matrices determine the behavior of the CNN (Biey, Checco, & Gilli, 2003). The templates specify how an image $\{f(x_i, y_j) : 1 \leq i \leq M, 1 \leq j \leq N : M, N \in \mathbb{Z}^+\}$ at time $t = 0$ will be transformed to produce an $M \times N$ output image $y(t)$ for $t \geq 0$. The outputs of the segmentation stage are then passed on to feature extraction modules to be used by fourier descriptors. A sample edge output for the genetically optimized templates in Equation 4.1.2 is shown in Figure 4.6. The configuration for the GA is shown in Table 4.1. The chromosome length is 19 representing $3 \times 3 + 3 \times 3 + 1$ of feedforward and feedback templates with the bias term as shown in the CNN equation.

Table 4.1: Configuration for the GA

GA Parameter	Value
Population size	50
Genomelength	19
Population type	real
Fitness Function	function f(.) in equation 4.1.1
No of generations	100
No of GA Iteration	1
Crossover	Heuristic Crossover
Crossover Fraction	0.8
Mutation	Uniform Mutation
Mutation Fraction	0.01

Selection scheme	Tournament of size 2
EliteCount	2

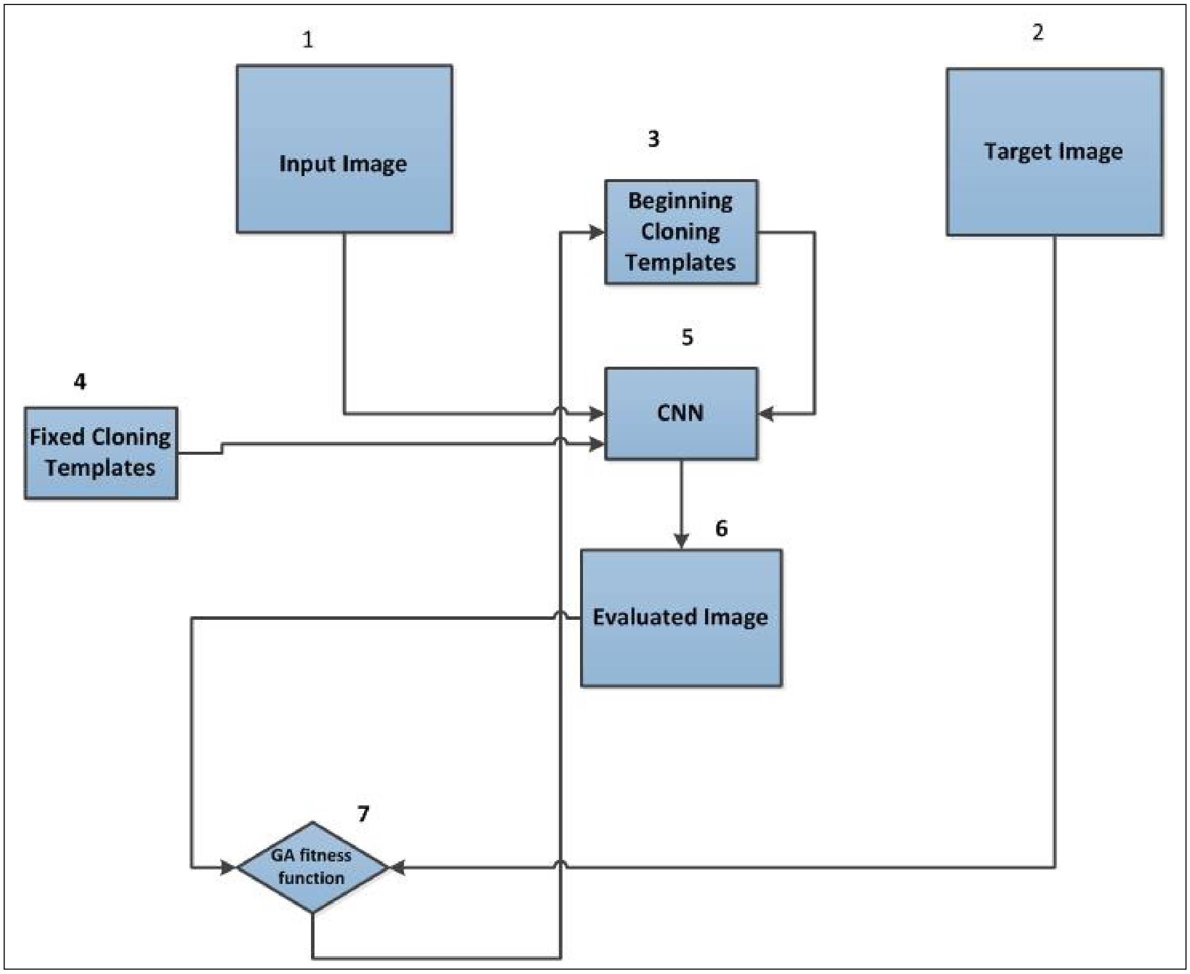
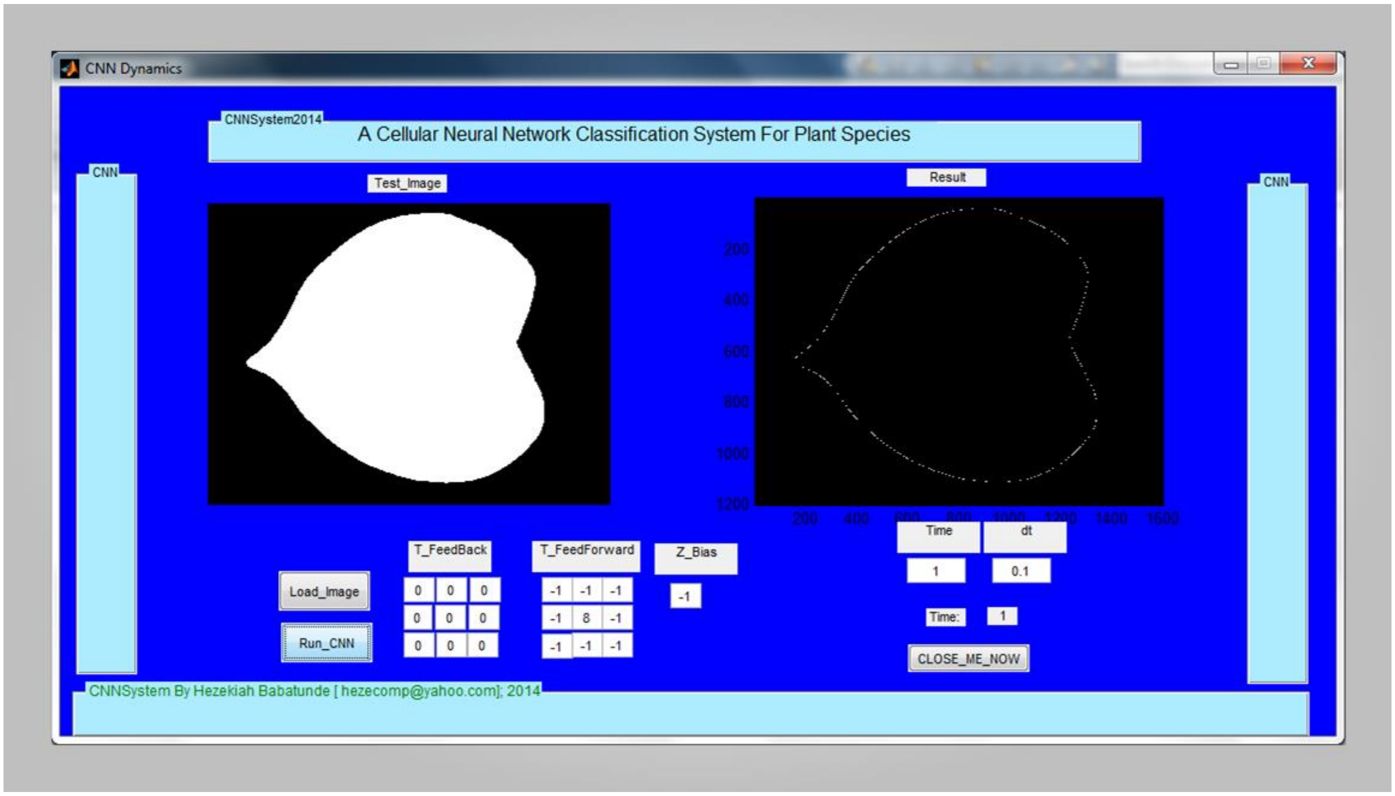


Figure 4.4: Genetic Cellular Neural Networks



A Cellular Neural Network Classification System For Plant Species

T_FeedBack

0	0	0
0	0	0
0	0	0

T_FeedForward

-1	-1	-1
-1	8	-1
-1	-1	-1

Z_Bias -1

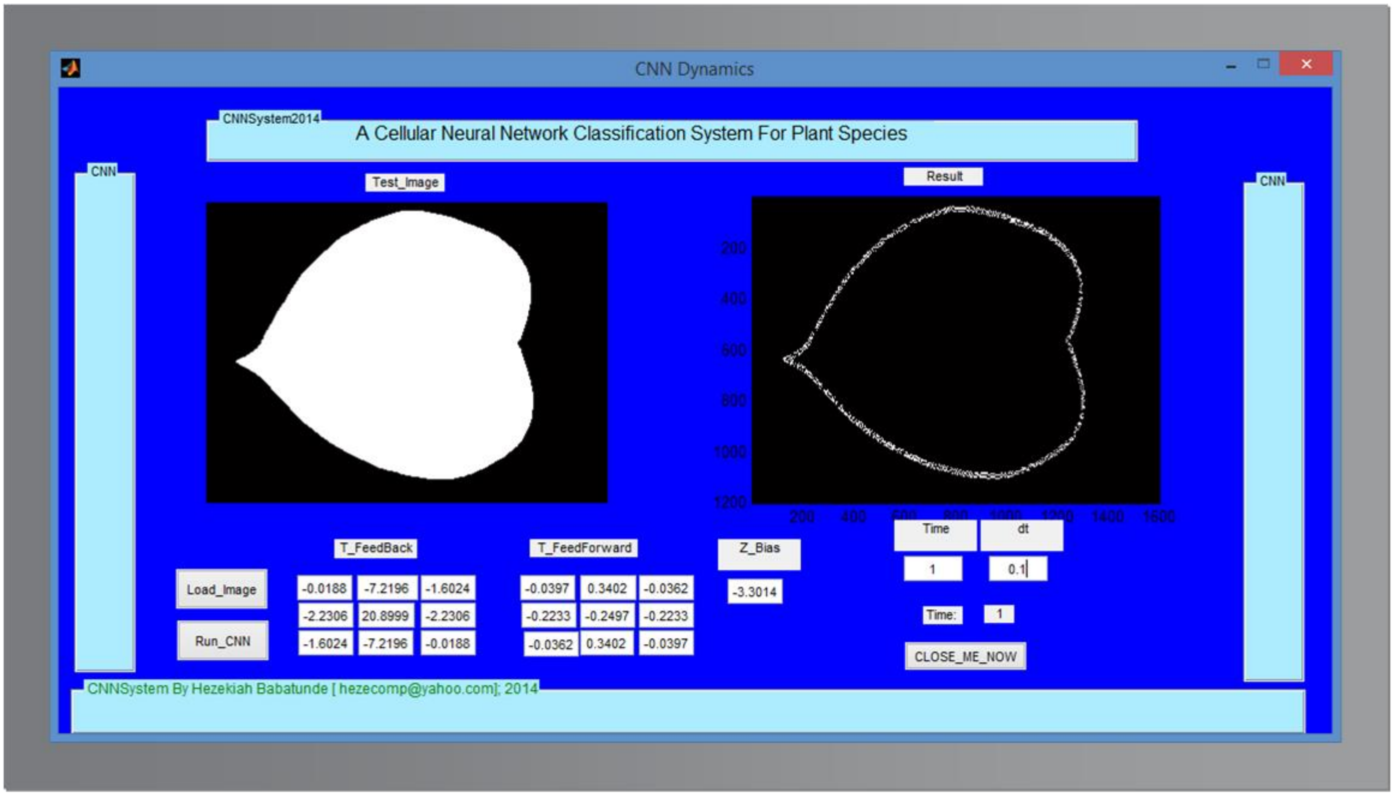
Time 1

dt 0.1

Time: 1

CNNSystem By Hezekiah Babatunde [hezecomp@yahoo.com]; 2014

Figure 4.5: CNN system for edge detection: ordinary templates



A Cellular Neural Network Classification System For Plant Species

T_FeedBack

-0.0188	-7.2196	-1.6024
-2.2306	20.8999	-2.2306
-1.6024	-7.2196	-0.0188

T_FeedForward

-0.0397	0.3402	-0.0362
-0.2233	-0.2497	-0.2233
-0.0362	0.3402	-0.0397

Z_Bias -3.3014

Time 1

dt 0.1

Time: 1

CNNSystem By Hezekiah Babatunde [hezecomp@yahoo.com]; 2014

Figure 4.6: Genetic CNN system for edge detection: genetic templates

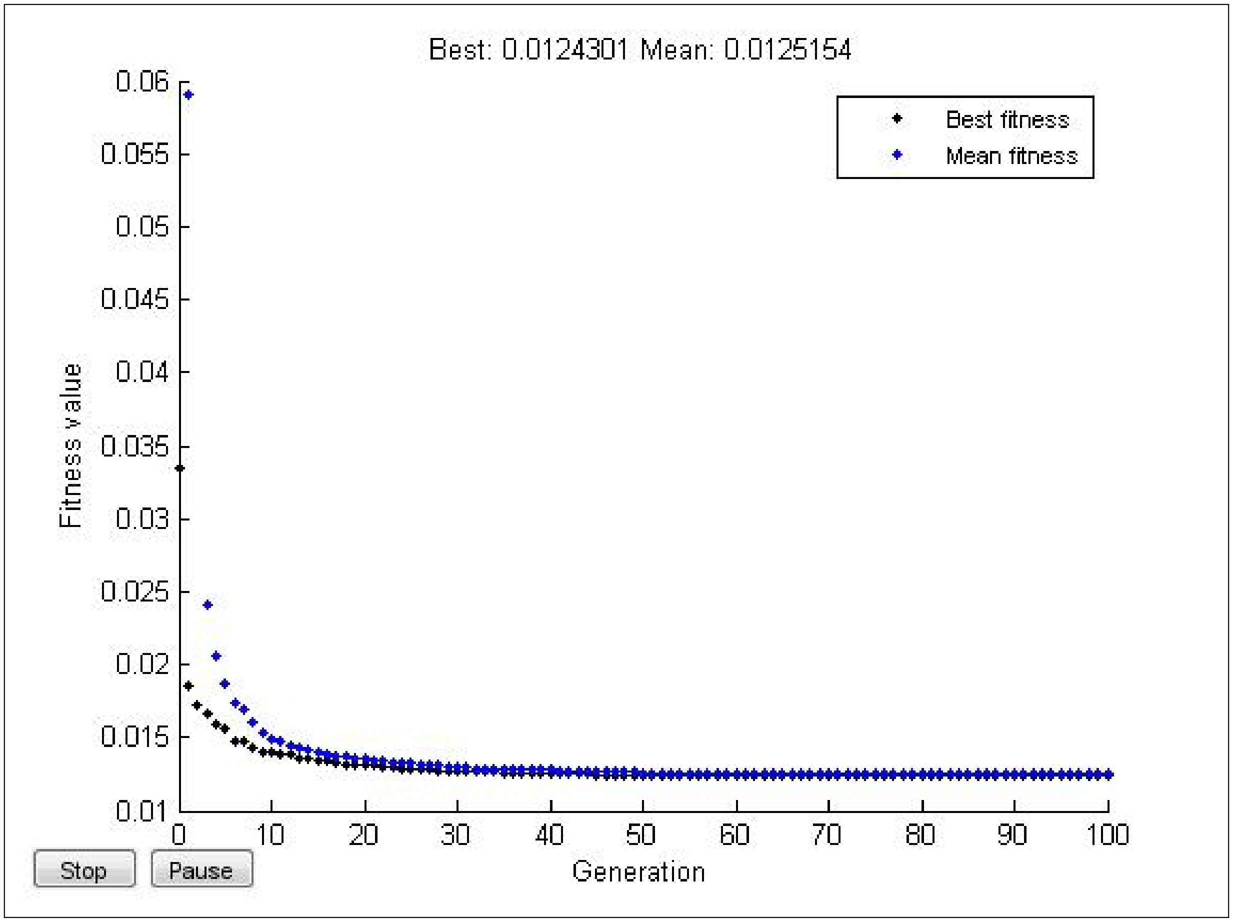


Figure 4.7: Fitness plot for the GA

4.1.3 Feature extraction

The features (image descriptors) is a set of scalar(s) associated with a digital image through which the image can be recognized. The descriptors can be invariant or variant to translation, rotation, and scaling (TRS) but an ideal descriptor should be invariant to TRS. The image descriptors (features) used in this study can be generally categorised into three types as shown in Figure 4.8. The following properties are associated with the descriptors used in the work: uniqueness (unique representation of object), completeness, affine mapping (invariance under translation, rotation, scaling, and reflection), sensitivity (reflecting differences between similar objects), and abstraction (robustness to noise).

The image descriptors used are Zernike Moments (ZM), Fourier Descriptors (FD), Lengendre Moments (LM), Hu 7 Moments (Hu7M), Texture Properties (TP) , Geometrical Properties (GP), and Colour features (CF). The features are fully explained in this section. The texture features include contrast, correlation, energy, homogenity and their variations. It should be noted that all these descriptors were real numbers which are mostly continous (see Figure 4.9) and can be normalized to be in the interval $[0, 1]$ or $[-1, 1]$ if need be.

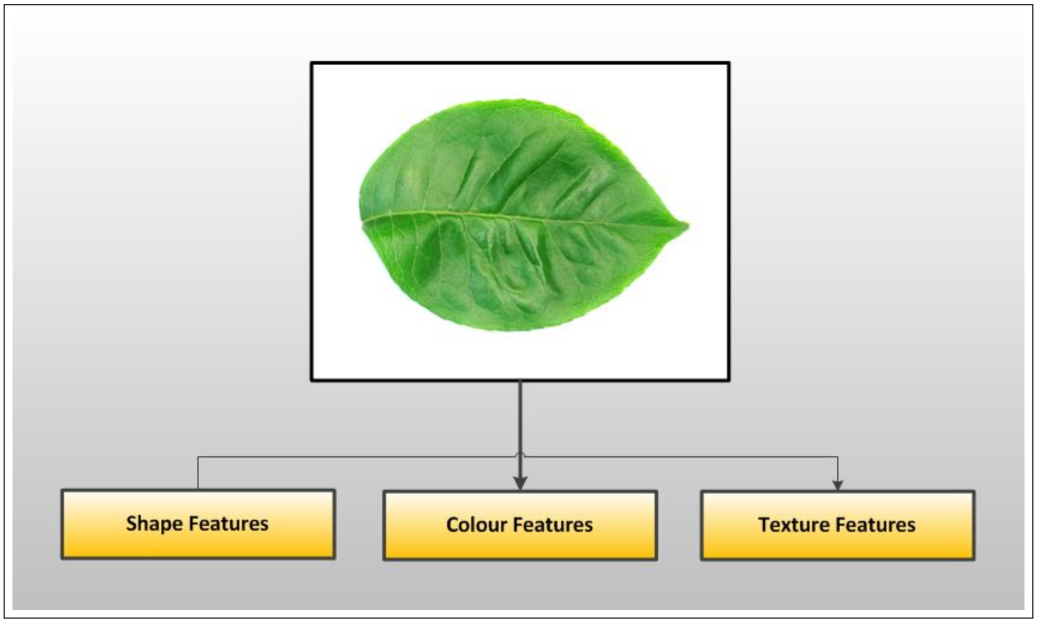
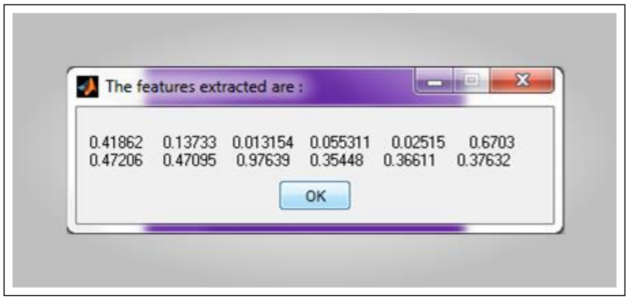


Figure 4.8: Generic content-based image descriptors



The features extracted are :

```
0.41862   0.13733   0.013154   0.055311   0.02515    0.6703
0.47206   0.47095   0.97639    0.35448    0.36611    0.37632
```

Figure 4.9: Examples of extracted features

4.1.3.1 Zernike Moments (Shape features)

The Zernike moment (ZM) can be defined as a set of complete complex orthogonal basis functions that are square integrable and that are defined over the unit disk (see Figure 4.11). An open disk around a given point, say, x in a plane is the set of all points in the plane whose distance from x is less than 1 (see Equation 4.1.4). For a closed disk, the distance from x is less than or equal to ρ where ($\rho = 1$ as shown in Equation 4.1.5).

$$D(x) = \{y \in \mathbb{R} : \|x - y\| < \rho\} \tag{4.1.4}$$

$$\bar{D}(x) = \{y \in \mathbb{R} : \|x - y\| \leq \rho\} \tag{4.1.5}$$

The steps involved in computing the ZMs for the dataset are shown in Figure 4.10.

1. Computation of Zernike Moments for Images of Plant leaves

   (a) STEP 1: The dimension of the original RGB leaf images in the dataset is $1200 \times 1600 \times 3$ containing different integers $\{Z_i, i = 1(1)3 : 1 \leq Z_i \leq 255\}$, representing RGB value for the leaf's image. After reading the RGB images into MATLAB workspace, we have the following numbers as output (represented as SET A)

   SET A = [76 75 77 82 88 93 95 93 79 76 73 71 71 75 81 84 81 83 85 86 83 81 83 85 94 109 117 113 105 99 87 75 79 80 83 88 91 90 87 82 74 71 68 66 66 70 74 77 68 73 79 83 81 77 74 76 101 112 118 110 101 93 82 7281 82 87 92 93 87 78 69 68 66 64 62 62 64 67 69 63 72 82 90 89 101 100 98 ...255].

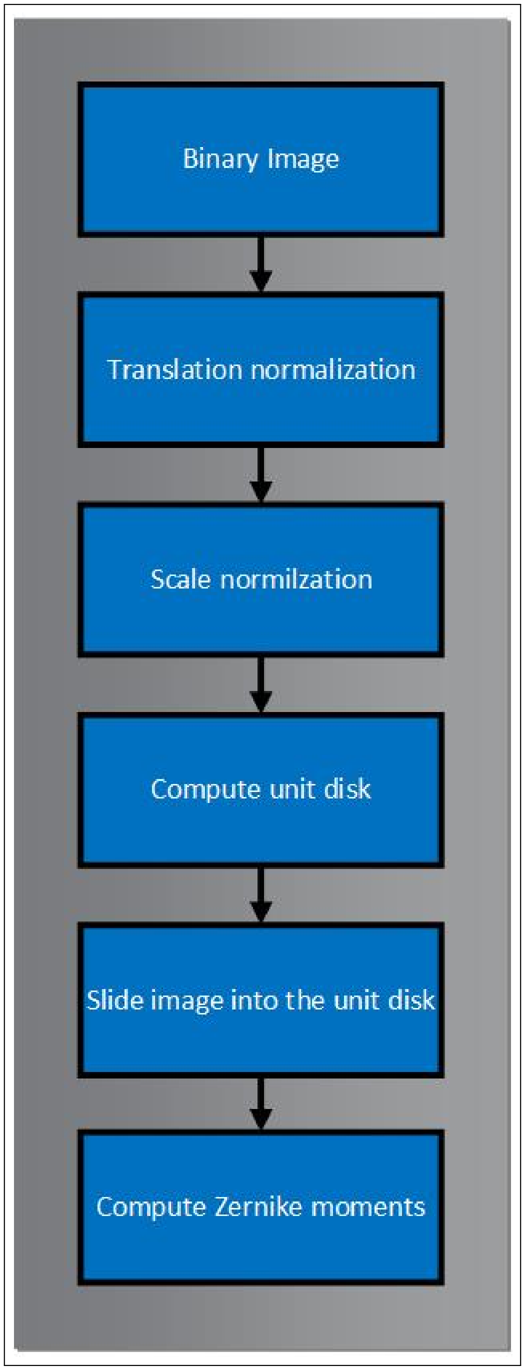


Figure 4.10: Computation of Zernike moments

Therefore n (SET A) = 1200 * 1600 * 3 = 5760000. In order words, n represents the total numbers used to represent the R,G,B values from the image.

(b) STEP 2 (Conversion to grayscale): The RGB images are converted to grayscale using appropriate formula as shown in Equation 2.4.1.

The size of each grayscale image in STEP 2 is 1200×1600. Thus, the total number of pixels here is 1200 x 1600 = 1920000. This implies the pixel values (intensity information) are spread across a rectangular or square region. This is a monochrome (or grayscale) image. The numerical output for this image is $\{Z_i\}_{i=1}^{1920000} : 1 \leq Z_i \leq 255$. Some of the pixel values are listed (as SET B) below:

SET B = [255 255 255 255 255 255 255 255 255 255 ...119 116 126 125 122 120 119 120 120 121 130 133 140 148 154 159 254 253 251 247 245 242 239 234 226 217 210 188 173 152 138 134 134 135 135 145 147 148 148.......255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255...].

Similarly, n (SET B) = 1200 * 1600 = 1920000. In other words, n represents the total numbers used to represent the intensities in the grayscale image of the leaf.

(c) STEP 3 (Conversion to binary image): This step further simplifies the greyscale image from STEP 2 to binary image to facilitate further computation. An effiecient way to do this conversion is to use the function im2bw() from MATLAB Toolbox. A thresholding function such as found in Otsu (1979) was used in MATLAB to achieve this (after grayscale conversion). Figure 4.12c represents a typical output from this step. The size of the binary image is still 1200×1600. This implies the total number of pixels information is still 1200 x 1600 = 1920000. The numerical output for this image is$\{Z_i, i = 1 : Z_i \in \{0,1\}\}$. Some of the pixel values are listed (as SET C) below:

SET C = [...1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...].

The dimension of SET B = dimension of SET C but the contents of SET C are bits which are easier for feature extractors and even computers to process.

2. Computational Algorithms and Image Normalization

Given the definitions in Equations 4.1.8 to 4.1.10, the Zernike moment, Z_{nm} for any given sample image $\{f(x_i, y_j) : 1 \leq i \leq M, 1 \leq j \leq N\}$ from the Flavia dataset, can be calculated as Equations (4.1.6) or (4.1.7)

$$Z_{nm} = \frac{n+1}{\pi} \int \int_{D} f(x,y) V^*_{nm}(x,y) dxdy = \frac{n+1}{\pi} \sum_{x}^{M} \sum_{y}^{N} V^*_{nm}(x,y) f(x,y) \tag{4.1.6}$$

where $x^2 + y^2 \leq 1$, and $m = 0, 1, 2, 3, ...\infty$. The m defines the order of the Zernike Polynomial while n which is either negative or positive, represents the multiplicity of the phase angles in ZM.

$$Z_{nm} = \frac{n+1}{\pi} \int_{0}^{2\pi} \int_{0}^{1} f(\rho, \theta) R_{nm}(\rho) e^{-im\theta} \rho d\rho d\theta \tag{4.1.7}$$

$$V_{nm}(\rho, \theta) = R_{nm}(\rho) e^{im\theta}, \theta \leq 1 \tag{4.1.8}$$

where

$$\rho = \sqrt{x^2 + y^2}, \theta = \arctan(\frac{y}{x}) \tag{4.1.9}$$

are the image pixel radial vector and angle between it and x-axis respectively

$$R_{nm}(\rho) = \sum_{a=0}^{\frac{(n-|m|)}{2}} (-1)^a \frac{(n-a)!}{a!(\frac{(n+|m|)}{2} - a)!(\frac{(n-|m|)}{2} - a)!} \rho^{n-2a} \tag{4.1.10}$$

The R_{nm} is the Zernike radial basis polynomial, some of which are listed in Table 9.1. The following conditions must be satisfied:

$$(a) n \in Z^{+}$$

$$(b) n - |m| \quad is\ \ even$$

$$(c) |m| \leq n$$

$$(d) \int_{0}^{2\pi} \int_{0}^{1} V_{nm}^{*}(\rho, \theta) \rho d\rho d\theta = \frac{\pi}{n+1} \delta_{np} \delta_{mq}, \delta_{zv} = \begin{cases} 1 & \text{z=v ,} \\ 0, & otherwise \end{cases} \tag{4.1.11}$$

A notable numerical property of Zernike polynomial is that they are always in the range −1 to +1 as given in the following expression:

$$|Z_{n}^{m}(\rho, \theta) = Z_{nm}(\rho, \theta)| \leq 1 \tag{4.1.12}$$

The results from the assertions from Equation 4.1.12 are shown in Table 4.2. Depending on the values of m the polynomials are either odd or even as seen in the following expression:

$$Z_{n}^{m}(\rho, \theta) = Z_{nm}(\rho, \theta) = R_{n}^{m}(\rho) \cos(m\theta) = R_{mn}(\rho) \cos(m\theta) \tag{4.1.13}$$

$$Z_{n}^{-m}(\rho, \theta) = Z_{-m,n}(\rho, \theta) = R_{n}^{-m}(\rho) \sin(m\theta) = R_{-m,n}(\rho) \sin(m\theta) \tag{4.1.14}$$

The problem here is to slide pixel information for the binary image from rectangular or square grid over a unit disk as shown in Figure 4.11. The ZM can be computed using cartesian coordinate system and polar coordinate system.

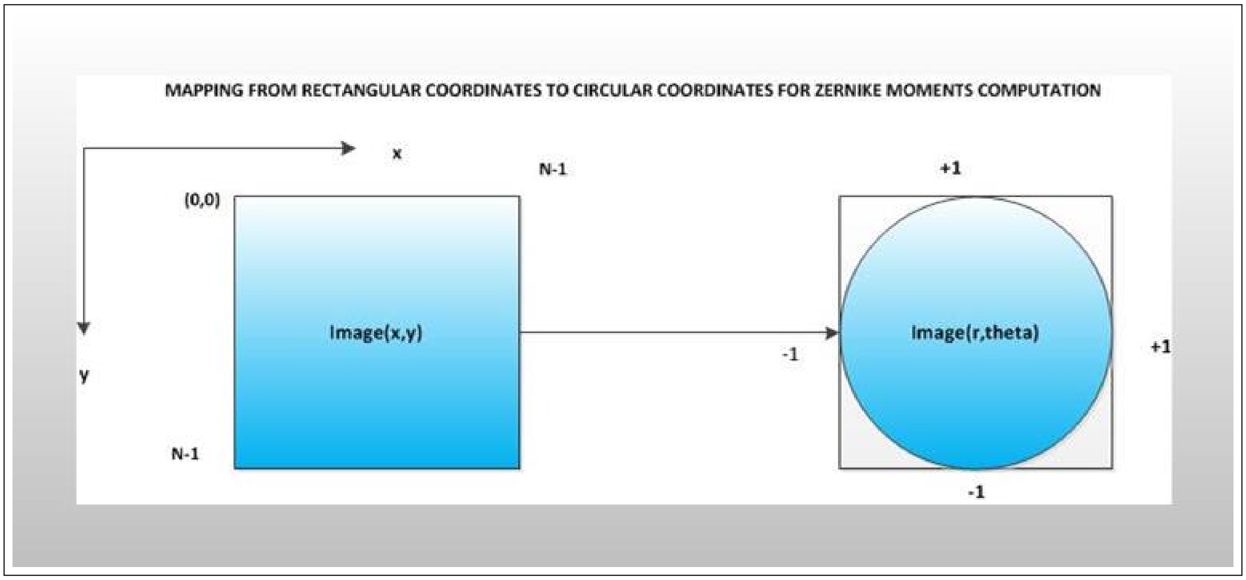

MAPPING FROM RECTANGULAR COORDINATES TO CIRCULAR COORDINATES FOR ZERNIKE MOMENTS COMPUTATION
x
y
(0,0)
N-1
N-1
Image(x,y)
+1
-1
+1
-1
Image(r,theta)

Figure 4.11: Conversion from rectangular to polar coordinates

Cartesian Coordinate System

Given a digital image $f(x_i, y_j)$, ,$i = 1(1)N$, ,$j = 1(1)N$. The xy-space is discretised as $x_i = (2i - N - 1)/N$ and $y_j = 2j - N - 1)/N$, where the double summation is performed over the (i, j) pairs that satisfy Equation 4.1.6. The function represents the image being described (in this case ROI from a plant leaf image). Integer m is either positive or negative, depicting the angular dependence or rotation, subject to the conditions (b) and (c) of Equation 4.1.10. The asterisk over the function V means complex conjugate.

Polar Coordinate System

The ROI is mapped to the unit disc (using Eq 4.1.9) through polar coordinates, where the center of the ROI is the origin of the unit disk. The conversion from rectangular to polar coordinates is done through Eq (4.1.9). The coordinates are then described by the length of the vector from the origin of the disk to the coordinate point ρ , and the angle from the x-axis, to the vector ρ, (the polar radius). The polar angle is represented as θ. The pixels falling outside the unit disc are not used in the calculation. The translation invariance is achieved by moving the centroid of the ROI to the origin of the disk and this eventually causes $m_{01} = m_{10} = 0$. The centroid of the ROI is given by the coordinates $(\bar{x},\bar{y})$ where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}} \tag{4.1.15}$$

The scale invariance for ZM is achieved through normalization of the image so that the total area of the forground pixels is of predetermined value, say, β. If the scaled version of the image $f(x,y)$ is represented as $f(x/a,y/a)$, the regular moment m_{pq} of $f(x,y)$ and m^1_{pq} of $f(x/a,y/a)$ are related by:

$$m^1_{pq} = \int_x \int_y x^p y^q f(\frac{x}{a},\frac{y}{a}) dxdy \tag{4.1.16}$$

$$= \int_x \int_y a^{p+q+2} x^p y^q f(x,y) a^2 dxdy \tag{4.1.17}$$

$$= a^{p+q+2} \int_x \int_y f(x,y) dxdy \tag{4.1.18}$$

$$= a^{p+q+2} m_{pq} \tag{4.1.19}$$

The target herein is to have $m^1_{00} = \beta$ and so, let

$$a = \sqrt{\frac{\beta}{m_{00}}} \tag{4.1.20}$$

and then substitute for a in m^1_{00} to have $m^1_{00} = a^2 m_{00} = \beta$. Therefore, translation and scaling invariance is achieved through the formula in Equation 4.1.21.

$$g(x,y) = f\big(\frac{x}{a} + \bar{x}, \frac{y}{a} + \bar{y}\big) \tag{4.1.21}$$

where

$$a = \sqrt{\frac{\beta}{m_{00}}} \tag{4.1.22}$$

3. Results of feature extraction based on ZM

The computation results showed that the numerical values of the ZM with respect to translation, rotation, and scalings are invariant or of negligble differences. The numerical values for the ZM computation are shown in Table 4.2. The rotations, scalings, and translation were taken over {15, 35, 45, 60}, {0.25, 0.5, 0.75}, and {(1,2)}, repectively. The results in Table 4.2 represent the ZM values computed on the first twenty set of images taken from the Flavia dataset (Wu et al., 2007). Figure 4.12 showed the results of ZM computation over original ROI, rotated ROI at angle 30, scaled ROI (by 0.75) and translated ROI (at (1,2)). Similarly, Figures 4.16 (a) and (b) showed the graph of ZM amplitude plotted against angles of rotation and scaling values respectively. Figure 4.14 also shows the amplitude of the ZM with respect to original, rotated, scaled, and translated image respectively. These two plots show that the computation of ZM of the same order and repetition across different angles, scalings and translation of the original ROI is constant. A re-constructed version of the original binary image of a leaf are shown (see Figure 4.17) across different orders (orders 5 to 60) of ZM showing the accuracy of ZM computation in this research. The figure shows that order 60 is sufficient to reconstruct original images. The rotational properties for the ZM has been proved in the appendix section of the thesis (see Equation 9.4.6). The first 33 instances of the first 14 Zernike Moments extracted from the images of plants' leaves in the Flavia dataset are shown in Figure 4.18.

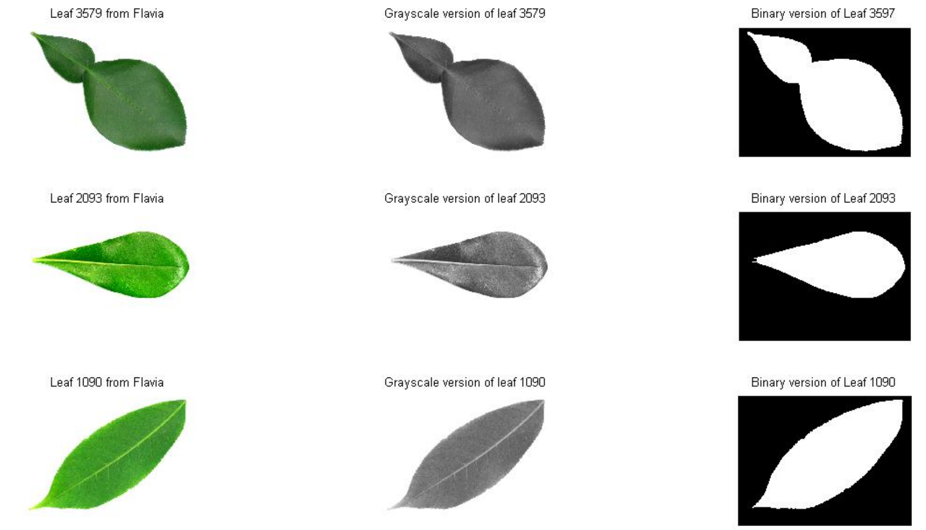

Leaf 3579 from Flavia
Grayscale version of leaf 3579
Binary version of Leaf 3597
Leaf 2093 from Flavia
Grayscale version of leaf 2093
Binary version of Leaf 2093
Leaf 1090 from Flavia
Grayscale version of leaf 1090
Binary version of Leaf 1090

Figure 4.12: Leaves of three species of plant taken from the Flavia dataset

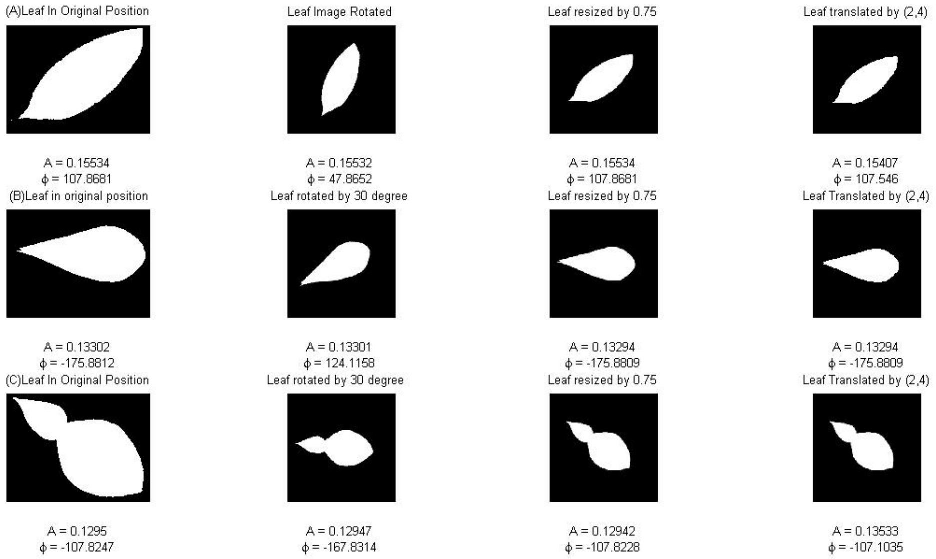


Figure 4.13: Invariance property of ZM under Translation, Rotation, and Scalings

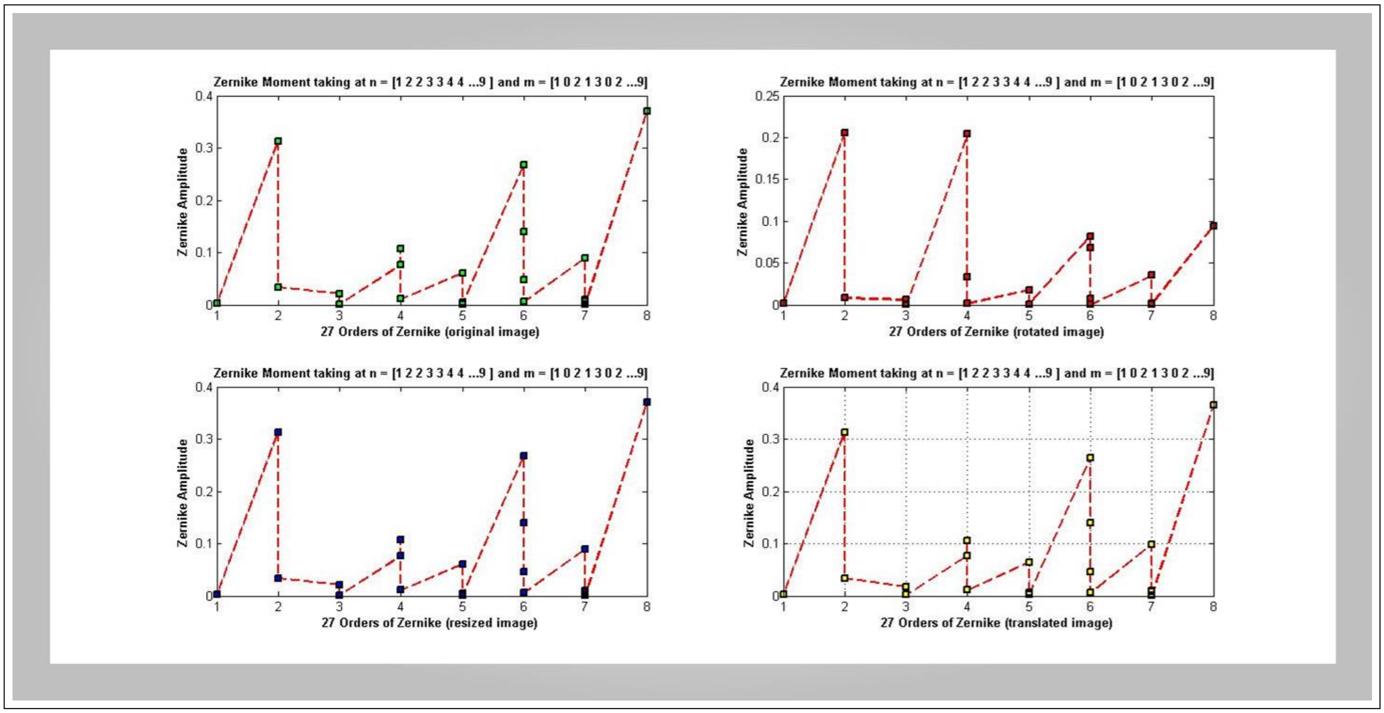


Figure 4.14: Invariance property of ZM under Translation, Rotation, and Scaling

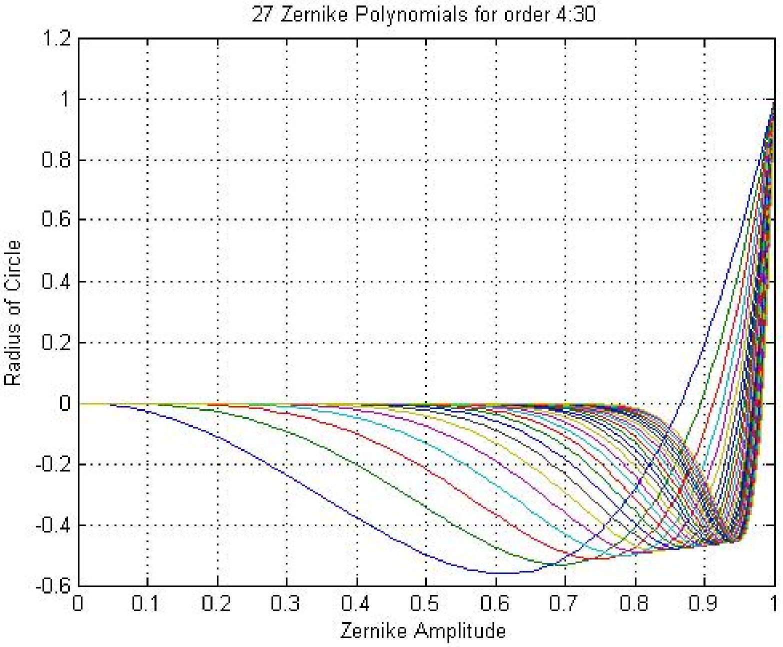


Figure 4.15: Zernike Polynomials for orders 4 to 30

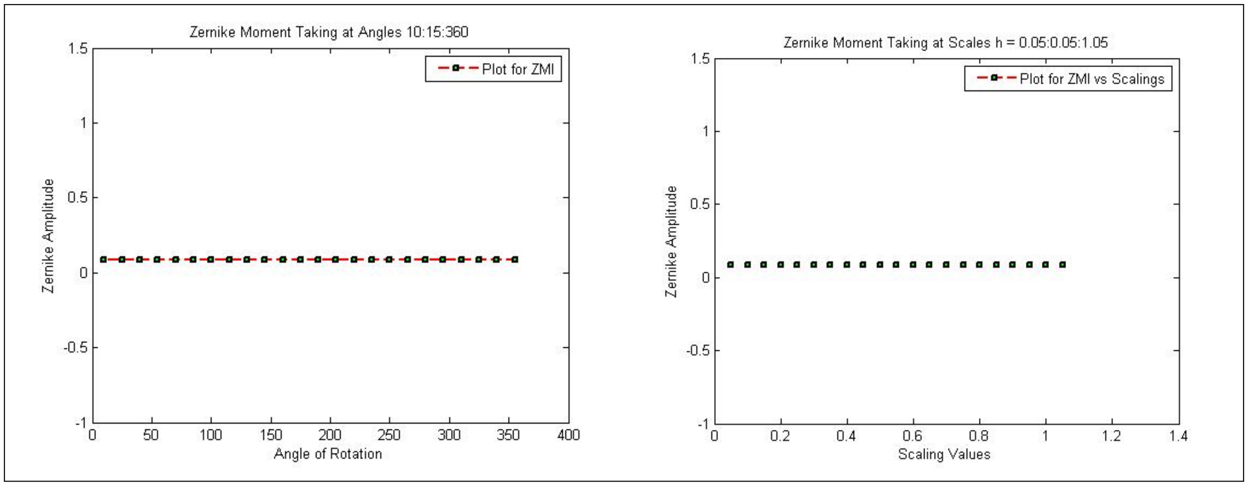


Figure 4.16: Zernike moment amplitude plot against angles and scalings

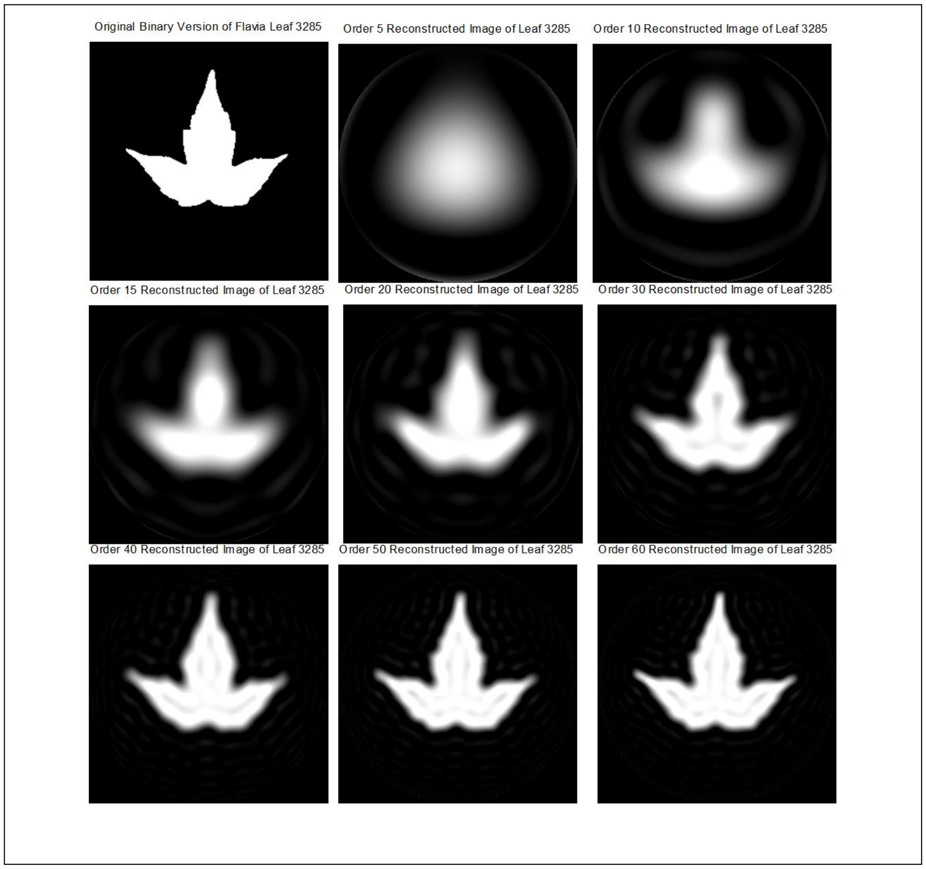
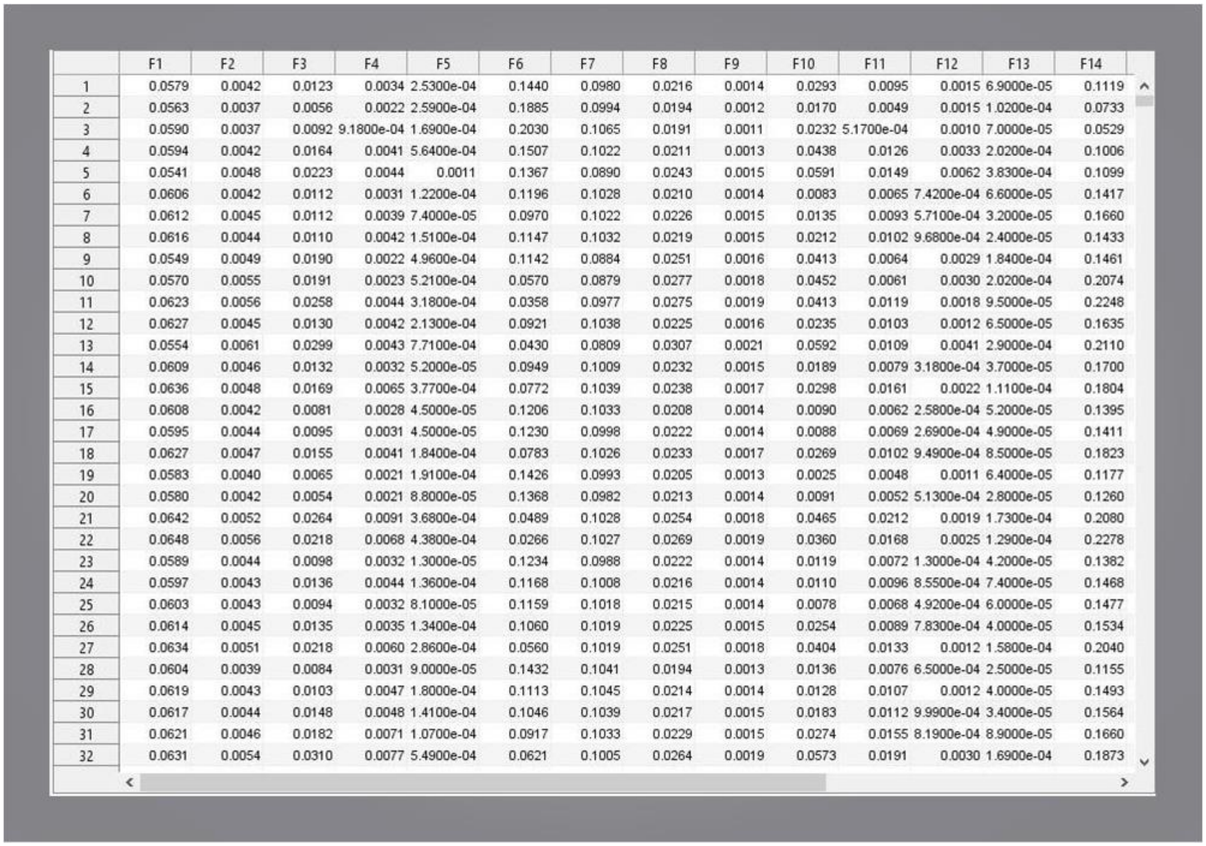

Original Binary Version of Flavia Leaf 3285
Order 5 Reconstructed Image of Leaf 3285
Order 10 Reconstructed Image of Leaf 3285
Order 15 Reconstructed Image of Leaf 3285
Order 20 Reconstructed Image of Leaf 3285
Order 30 Reconstructed Image of Leaf 3285
Order 40 Reconstructed Image of Leaf 3285
Order 50 Reconstructed Image of Leaf 3285
Order 60 Reconstructed Image of Leaf 3285

Figure 4.17: Original and Reconstructed Image

Table 4.2: Invariant ZM under Translation, Rotation, and Scaling

No	ZM	ZM15D	ZM35D	ZM45D	ZM60D	ZM0.25	ZM0.35	ZM0.5	ZM0.75	ZMT12	ZMT22
1	0.0579	0.0579	0.0579	0.0581	0.0579	0.0577	0.0569	0.0578	0.0579	0.0573	0.0574
2	0.0562	0.0562	0.0562	0.0563	0.0562	0.0560	0.0552	0.0560	0.0562	0.0557	0.0558
3	0.0589	0.0589	0.0589	0.0586	0.0589	0.0587	0.0579	0.0588	0.0589	0.0584	0.0587
4	0.0593	0.0593	0.0593	0.0591	0.0593	0.0591	0.0583	0.0592	0.0593	0.0588	0.0591
5	0.0540	0.0540	0.0540	0.0540	0.0540	0.0540	0.0529	0.0539	0.0539	0.0538	0.0541
6	0.0605	0.0605	0.0605	0.0605	0.0605	0.0603	0.0594	0.0604	0.0605	0.0605	0.0605
7	0.0611	0.0612	0.0612	0.0612	0.0611	0.0610	0.0600	0.0610	0.0611	0.0610	0.0610
8	0.0603	0.0603	0.0603	0.0601	0.0603	0.0601	0.0592	0.0602	0.0603	0.0601	0.0601
9	0.0548	0.0548	0.0548	0.0548	0.0548	0.0546	0.0539	0.0547	0.0548	0.0544	0.0544
10	0.0569	0.0569	0.0569	0.0569	0.0569	0.0567	0.0560	0.0568	0.0569	0.0565	0.0565
11	0.0623	0.0623	0.063	0.0620	0.0623	0.0621	0.0613	0.0622	0.0622	0.0619	0.0616
12	0.0614	0.0614	0.0614	0.0615	0.0614	0.0611	0.0603	0.0613	0.0613	0.0613	0.0612
13	0.0554	0.0554	0.0554	0.0556	0.0554	0.0552	0.0546	0.0553	0.0553	0.0550	0.0548
14	0.0609	0.0608	0.0609	0.0609	0.0608	0.0606	0.0598	0.0608	0.0608	0.0607	0.0607
15	0.0636	0.0636	0.0636	0.0634	0.0636	0.0633	0.0625	0.0634	0.0636	0.0632	0.0631
16	0.0635	0.0635	0.0635	0.0634	0.0635	0.0632	0.0624	0.0633	0.0634	0.0632	0.0631
17	0.0594	0.0594	0.0594	0.0594	0.0594	0.0592	0.0584	0.0593	0.0594	0.0592	0.0592
18	0.0627	0.0627	0.0627	0.0629	0.0627	0.0624	0.0616	0.0625	0.0626	0.0624	0.0623
19	0.0583	0.0582	0.0582	0.0582	0.0582	0.0580	0.0572	0.0581	0.0582	0.0579	0.0579
20	0.0580	0.05780	0.0580	0.0579	0.0580	0.0577	0.0569	0.0578	0.0580	0.0576	0.0577



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
1	0.0579	0.0042	0.0123	0.0034	2.5300e-04	0.1440	0.0980	0.0216	0.0014	0.0293	0.0095	0.0015	6.9000e-05	0.1119
2	0.0563	0.0037	0.0056	0.0022	2.5900e-04	0.1885	0.0994	0.0194	0.0012	0.0170	0.0049	0.0015	1.0200e-04	0.0733
3	0.0590	0.0037	0.0092	9.1800e-04	1.6900e-04	0.2030	0.1065	0.0191	0.0011	0.0232	5.1700e-04	0.0010	7.0000e-05	0.0529
4	0.0594	0.0042	0.0164	0.0041	5.6400e-04	0.1507	0.1022	0.0211	0.0013	0.0438	0.0126	0.0033	2.0200e-04	0.1006
5	0.0541	0.0048	0.0223	0.0044	0.0011	0.1367	0.0890	0.0243	0.0015	0.0591	0.0149	0.0062	3.8300e-04	0.1099
6	0.0606	0.0042	0.0112	0.0031	1.2200e-04	0.1196	0.1028	0.0210	0.0014	0.0083	0.0065	7.4200e-04	6.6000e-05	0.1417
7	0.0612	0.0045	0.0112	0.0039	7.4000e-05	0.0970	0.1022	0.0226	0.0015	0.0135	0.0093	5.7100e-04	3.2000e-05	0.1660
8	0.0616	0.0044	0.0110	0.0042	1.5100e-04	0.1147	0.1032	0.0219	0.0015	0.0212	0.0102	9.6800e-04	2.4000e-05	0.1433
9	0.0549	0.0049	0.0190	0.0022	4.9600e-04	0.1142	0.0884	0.0251	0.0016	0.0413	0.0064	0.0029	1.8400e-04	0.1461
10	0.0570	0.0055	0.0191	0.0023	5.2100e-04	0.0570	0.0879	0.0277	0.0018	0.0452	0.0061	0.0030	2.0200e-04	0.2074
11	0.0623	0.0056	0.0258	0.0044	3.1800e-04	0.0358	0.0977	0.0275	0.0019	0.0413	0.0119	0.0018	9.5000e-05	0.2248
12	0.0627	0.0045	0.0130	0.0042	2.1300e-04	0.0921	0.1038	0.0225	0.0016	0.0235	0.0103	0.0012	6.5000e-05	0.1635
13	0.0554	0.0061	0.0299	0.0043	7.7100e-04	0.0430	0.0809	0.0307	0.0021	0.0592	0.0109	0.0041	2.9000e-04	0.2110
14	0.0609	0.0046	0.0132	0.0032	5.2000e-05	0.0949	0.1009	0.0232	0.0015	0.0189	0.0079	3.1800e-04	3.7000e-05	0.1700
15	0.0636	0.0048	0.0169	0.0065	3.7700e-04	0.0772	0.1039	0.0238	0.0017	0.0298	0.0161	0.0022	1.1100e-04	0.1804
16	0.0608	0.0042	0.0081	0.0028	4.5000e-05	0.1206	0.1033	0.0208	0.0014	0.0090	0.0062	2.5800e-04	5.2000e-05	0.1395
17	0.0595	0.0044	0.0095	0.0031	4.5000e-05	0.1230	0.0998	0.0222	0.0014	0.0088	0.0069	2.6900e-04	4.9000e-05	0.1411
18	0.0627	0.0047	0.0155	0.0041	1.8400e-04	0.0783	0.1026	0.0233	0.0017	0.0269	0.0102	9.4900e-04	8.5000e-05	0.1823
19	0.0583	0.0040	0.0065	0.0021	1.9100e-04	0.1426	0.0993	0.0205	0.0013	0.0025	0.0048	0.0011	6.4000e-05	0.1177
20	0.0580	0.0042	0.0054	0.0021	8.8000e-05	0.1368	0.0982	0.0213	0.0014	0.0091	0.0052	5.1300e-04	2.8000e-05	0.1260
21	0.0642	0.0052	0.0264	0.0091	3.6800e-04	0.0489	0.1028	0.0254	0.0018	0.0465	0.0212	0.0019	1.7300e-04	0.2080
22	0.0648	0.0056	0.0218	0.0068	4.3800e-04	0.0266	0.1027	0.0269	0.0019	0.0360	0.0168	0.0025	1.2900e-04	0.2278
23	0.0589	0.0044	0.0098	0.0032	1.3000e-05	0.1234	0.0988	0.0222	0.0014	0.0119	0.0072	1.3000e-04	4.2000e-05	0.1382
24	0.0597	0.0043	0.0136	0.0044	1.3600e-04	0.1168	0.1008	0.0216	0.0014	0.0110	0.0096	8.5500e-04	7.4000e-05	0.1468
25	0.0603	0.0043	0.0094	0.0032	8.1000e-05	0.1159	0.1018	0.0215	0.0014	0.0078	0.0068	4.9200e-04	6.0000e-05	0.1477
26	0.0614	0.0045	0.0135	0.0035	1.3400e-04	0.1060	0.1019	0.0225	0.0015	0.0254	0.0089	7.8300e-04	4.0000e-05	0.1534
27	0.0634	0.0051	0.0218	0.0060	2.8600e-04	0.0560	0.1019	0.0251	0.0018	0.0404	0.0133	0.0012	1.5800e-04	0.2040
28	0.0604	0.0039	0.0084	0.0031	9.0000e-05	0.1432	0.1041	0.0194	0.0013	0.0136	0.0076	6.5000e-04	2.5000e-05	0.1155
29	0.0619	0.0043	0.0103	0.0047	1.8000e-04	0.1113	0.1045	0.0214	0.0014	0.0128	0.0107	0.0012	4.0000e-05	0.1493
30	0.0617	0.0044	0.0148	0.0048	1.4100e-04	0.1046	0.1039	0.0217	0.0015	0.0183	0.0112	9.9900e-04	3.4000e-05	0.1564
31	0.0621	0.0046	0.0182	0.0071	1.0700e-04	0.0917	0.1033	0.0229	0.0015	0.0274	0.0155	8.1900e-04	8.9000e-05	0.1660
32	0.0631	0.0054	0.0310	0.0077	5.4900e-04	0.0621	0.1005	0.0264	0.0019	0.0573	0.0191	0.0030	1.6900e-04	0.1873

Figure 4.18: The first 33 instances of the first 14 Zernike Moments extracted from the images of plants' leaves in the Flavia dataset

4.1.3.2 Fourier Descriptors (Shape features)

Fourier Descriptors (FD) were also extracted from the Flavai dataset. FDs are boundary-based feature extractor which employs application of Fourier theory to shape analysis or description (Nixon & Aguado, 2012). FD methods have been traditionally used for shape recognition and are part of general methods used in encoding various shape signatures (Charles & Ralph, 1972; Dengsheng & Guojun, 2000; Tyler, 2006). Particularly, Peters (2011) defined the FT as the decomposition of a nonperiodic signal into continous aggregation of sinusoids. The convolution operation used by Fourier descriptors is the Fourier transform (FT), which is a mathematical representation of a signal or digital image (say plant's leaf image) is a summation of complex exponentials containing varying magnitudes, frequencies, and phases (R. C. Gonzalez et al., 2009). The idea behind FT in image processing is to characterise a contour by a set of scalars which are representing the frequency content of the whole shape. Fourier descriptors (FD) method has been traditionally used for shape recognition and are part of general methods used in encoding various shape signatures (Charles & Ralph, 1972; Dengsheng & Guojun, 2000; Tyler, 2006). For an image $f(x,y)$ with $x = 0,1,2,...,M-1; y = 0,1,2,...,N-1$, the two-dimensional discrete FT (2D DFT) and its inverse (2D IDFT) are defined by Equations 4.1.23 & 4.1.24 (Fourier, 1878; MathWorks, 2007; Luminita, 2010).

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-2\pi i(\frac{ux}{M}+\frac{vy}{N})}, u = 0(1)M-1, v = 0(1)N-1. \tag{4.1.23}$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{2\pi i(\frac{ux}{M}+\frac{vy}{N})}, x = 0(1)M-1, y = 0(1)N-1. \tag{4.1.24}$$

The original image of the plant's leaf can be re-constructed by using the inverse Fourier Transform. To recreate the original image, all the fundamentals and harmonics are added together with each term weighted by its corresponding transform coefficient. FT is popular in image processing applications such as analysis, enhancement, restoration, and compression, since FT is a good candidate to achieve rotational invariance among shapes of images (Tyler, 2006; Onkar, 2011). The coefficients of Fourier series in Equation 4.1.23 are called Fourier descriptors. The leaf shape is analyzed in the frequency domain, rather than the spatial domain (Cope et al., 2012). The higher the harmonics of the FD, the higher the description precision.

Figure 4.19: A typical representation of Fourier Transform or Analysis

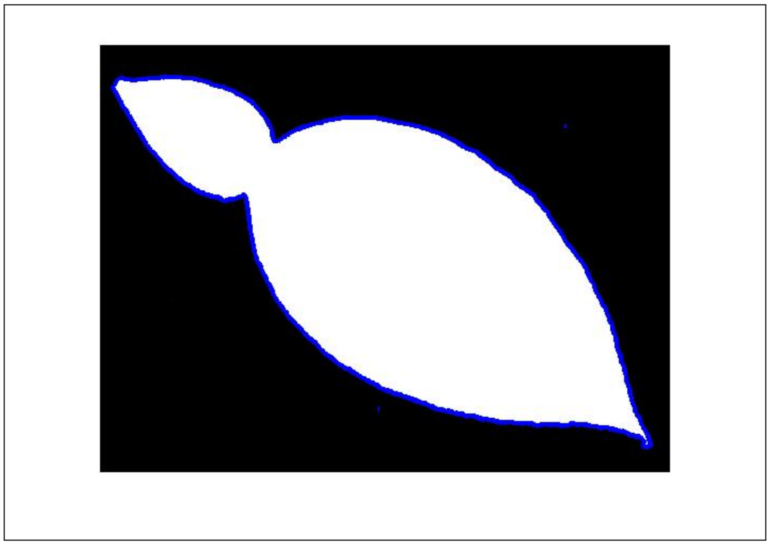


Figure 4.20: Leaf boundary

The steps needed (see Figure 4.21) to compute FDs are given as follows:

1. Boundary Parametization

   The boundaries of the image (say image in Figure 4.20) is given as the set $\big((x_1,y_1),(x_2,y_2),(x_3,y_3),...(x_N,y_N)\big)$ containing N ordered points/pixels.

2. Boundary Tracing

   Any arbitrary point, say, (x_0,y_0), is chosen as the starting point and the traversal (or tracing) of all the boundaries to the left and right of point (x_0,y_0). A complete list of boundary pixels will be obtained from this.

3. Complex representation of boundary points: The boundary pixels set $(x_i,y_j), i=0(1)N, j=1(1)N$, is then represented as complex variables $(x_n+jy_m), n=0(1)N, m=1(1), j^2=-1$

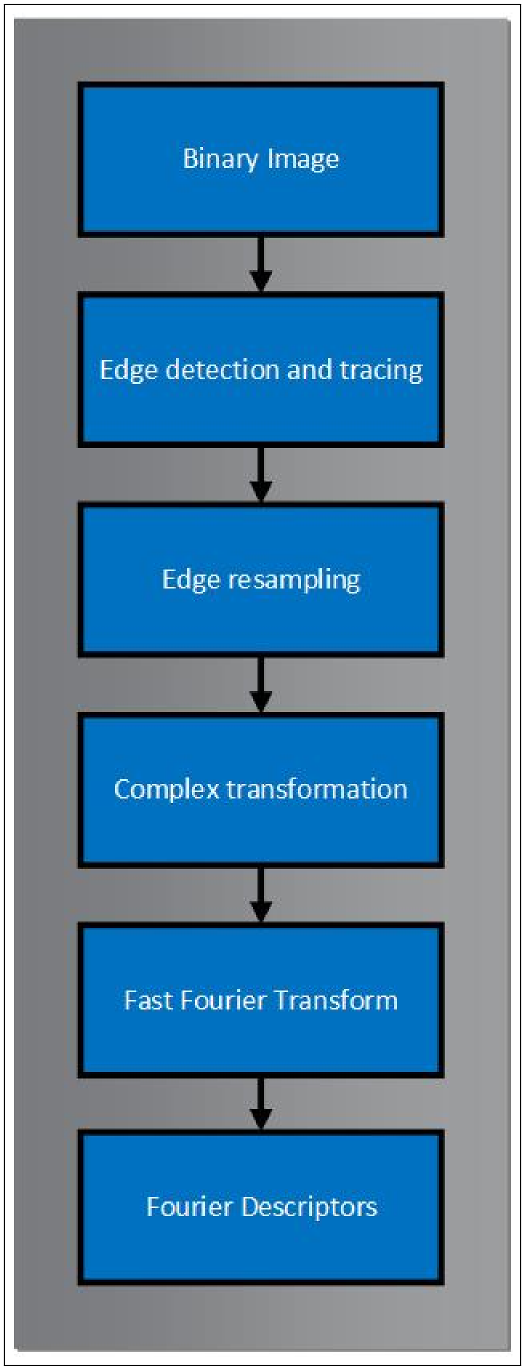


Figure 4.21: Algorithmic approach to computation of Fourier Descriptors

4. Application of Fourier Transform

   Next to complex representation above, appropriate Fourier Transform such as FFT or DFT (Mathworks, 2009) is applied to it. The coefficient thus obtained are called Fourier Descriptors.

5. Invariant FDs Suppose the descriptors are given as $|FD_1|, |FD_2|, |FD_3|, ..., |FD_N|$. To achieve invariant properties with respect to rotation, scaling, and translation, the following steps were performed:

   (a) Set $|FD_1| = 0$

   (b) Divide the remaining FDs by the first of the remaining FDs. This implies that

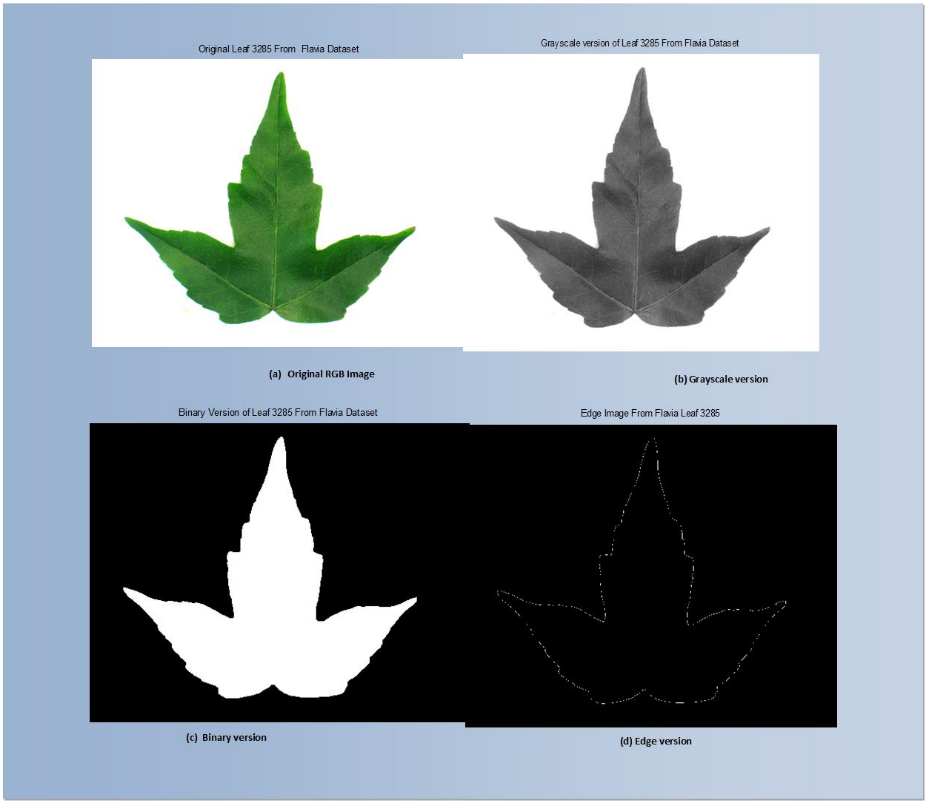
$$\begin{aligned}
|FD_2| &\mapsto \tfrac{|FD_2|}{|FD_2|}, \\
|FD_3| &\mapsto \tfrac{|FD_3|}{|FD_2|}, \\
|FD_4| &\mapsto \tfrac{|FD_4|}{|FD_2|}, \\
&\vdots \\
|FD_N| &\mapsto \tfrac{|FD_N|}{|FD_2|}
\end{aligned}$$

1. Geometric Implication of FDs

   The first element of FDs is called (the d.c component) and is simply the average value of the x and y coordinates, which are the coordinates of the center point of the boundary expressed as complex numbers. The second coefficient gives the radius of the circle that best fits the points. This implies that a circle can be described by its zero and first order components (i.e 0-harmonic and 1-harmonic respectively). Higher harmonics are used to show detailed image information.

2. Image Reconstruction Using Inverse Fourier Transforms

   Using some of the coefficients (not necessarily all), original boundary shape can be reconstructed. The reconstructed boundary shapes for leaf 3285 from the Flavia dataset, using 5, 10, 20, ...140 coefficients, are given in Figure 4.23


Original Leaf 3285 From Flavia Dataset
Grayscale version of Leaf 3285 From Flavia Dataset
Binary Version of Leaf 3285 From Flavia Dataset
Edge Image From Flavia Leaf 3285

(a) Original RGB Image

(b) Grayscale version

(c) Binary version

(d) Edge version

Figure 4.22: Original image, grayscale image, binary image and edge image of Leaf 3285 From Flavia Dataset

3. Results of feature extraction based on FD

   The results of these steps are given below. The results are shown for original, scaled, rotated and translated version of some set of images. The images were from the first 40 images taken from the Flavia dataset. A close inspection shows that ZMs are more invariant to TRS than FDs. Classification accuracy can be achieved using the FDs alone if more images are used in the training set. Image reconstruction through Fourier Descriptors using 5, 10, 20, ...140 coefficients is shown in Figure 4.23.

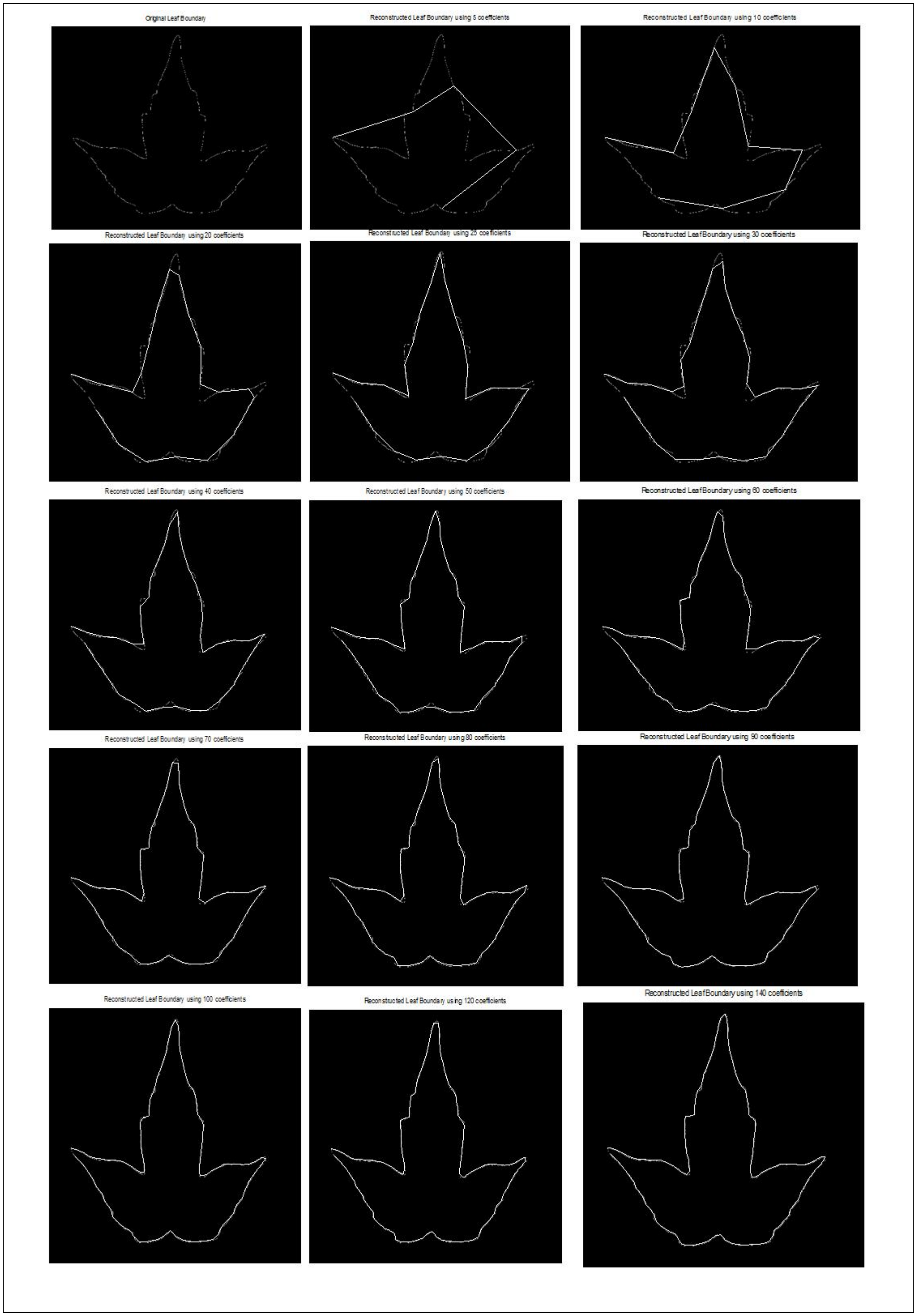
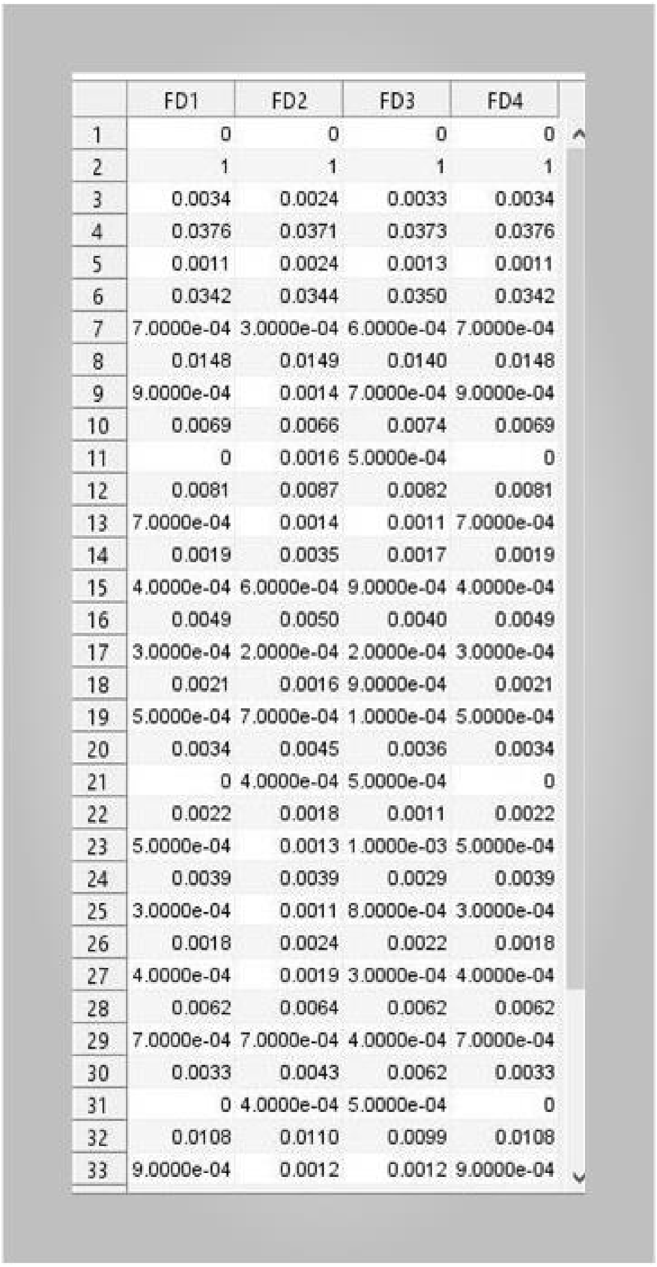

Original Leaf Boundary
Reconstructed Leaf Boundary using 5 coefficients
Reconstructed Leaf Boundary using 10 coefficients
Reconstructed Leaf Boundary using 20 coefficients
Reconstructed Leaf Boundary using 25 coefficients
Reconstructed Leaf Boundary using 30 coefficients
Reconstructed Leaf Boundary using 40 coefficients
Reconstructed Leaf Boundary using 50 coefficients
Reconstructed Leaf Boundary using 60 coefficients
Reconstructed Leaf Boundary using 70 coefficients
Reconstructed Leaf Boundary using 80 coefficients
Reconstructed Leaf Boundary using 90 coefficients
Reconstructed Leaf Boundary using 100 coefficients
Reconstructed Leaf Boundary using 120 coefficients
Reconstructed Leaf Boundary using 140 coefficients

Figure 4.23: Image Reconstruction through Fourier Descriptors using 5, 10, 20, ...140 coefficients.



	FD1	FD2	FD3	FD4
1	0	0	0	0
2	1	1	1	1
3	0.0034	0.0024	0.0033	0.0034
4	0.0376	0.0371	0.0373	0.0376
5	0.0011	0.0024	0.0013	0.0011
6	0.0342	0.0344	0.0350	0.0342
7	7.0000e-04	3.0000e-04	6.0000e-04	7.0000e-04
8	0.0148	0.0149	0.0140	0.0148
9	9.0000e-04	0.0014	7.0000e-04	9.0000e-04
10	0.0069	0.0066	0.0074	0.0069
11	0	0.0016	5.0000e-04	0
12	0.0081	0.0087	0.0082	0.0081
13	7.0000e-04	0.0014	0.0011	7.0000e-04
14	0.0019	0.0035	0.0017	0.0019
15	4.0000e-04	6.0000e-04	9.0000e-04	4.0000e-04
16	0.0049	0.0050	0.0040	0.0049
17	3.0000e-04	2.0000e-04	2.0000e-04	3.0000e-04
18	0.0021	0.0016	9.0000e-04	0.0021
19	5.0000e-04	7.0000e-04	1.0000e-04	5.0000e-04
20	0.0034	0.0045	0.0036	0.0034
21	0	4.0000e-04	5.0000e-04	0
22	0.0022	0.0018	0.0011	0.0022
23	5.0000e-04	0.0013	1.0000e-03	5.0000e-04
24	0.0039	0.0039	0.0029	0.0039
25	3.0000e-04	0.0011	8.0000e-04	3.0000e-04
26	0.0018	0.0024	0.0022	0.0018
27	4.0000e-04	0.0019	3.0000e-04	4.0000e-04
28	0.0062	0.0064	0.0062	0.0062
29	7.0000e-04	7.0000e-04	4.0000e-04	7.0000e-04
30	0.0033	0.0043	0.0062	0.0033
31	0	4.0000e-04	5.0000e-04	0
32	0.0108	0.0110	0.0099	0.0108
33	9.0000e-04	0.0012	0.0012	9.0000e-04

Figure 4.24: The first 33 FDs from the Flavia dataset in using scaled, rotated, and translated version of each images.

4.1.3.3 Hu 7 Moments (Shape features)

The seven moments proposed by Hu (1962) were also used as image descriptors in capturing the shape of leaves found in the Flavia dataset. These moments are invariant under translation, scaling, and rotation. Figure 4.25 shows the algorithmic approach to extraction of Hu's seven moments. Going from the binarized version of images in the Flavia dataset, image centroids were computed, followed by central moments and normalized moments. Finally, the Hu's seven moments were computed based on the formulars in Equations 4.1.25, 4.1.26, 4.1.27, 4.1.28, 4.1.29, 4.1.30, & 4.1.31.

$$\Phi_1 = \eta_{20} + \eta_{02}, \tag{4.1.25}$$

$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + (\eta_{11})^2, \tag{4.1.26}$$

$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + 3(\eta_{21} - \eta_{03})^2, \tag{4.1.27}$$

$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{4.1.28}$$

$$\begin{aligned}\Phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) + [(\eta_{21} + \eta_{03})^2 - 3(\eta_{21} + \eta_{03})^2] \\ + (\eta_{03} - 3\eta_{12})(\eta_{21} + \eta_{03})[(\eta_{12} + \eta_{03})^2 - 3(\eta_{12} + \eta_{30})]\end{aligned} \tag{4.1.29}$$

$$\Phi_6 = (\eta_{20} - \eta_{02})^2[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}) \tag{4.1.30}$$

$$\begin{aligned}\Phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] + \\ (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[(\eta_{03} + \eta_{21})^2 - 3(\eta_{12} + \eta_{30})^2]\end{aligned} \tag{4.1.31}$$

where

$$m_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^p y^q f(x,y)\,dx\,dy = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} x^i y^j f(x,y) \tag{4.1.32}$$

The central moments are given as:

$$\mu_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(x-\bar{x})^p(y-\bar{y})^q f(x,y)dxdy = \sum_x\sum_y(x-\bar{x})^p(y-\bar{y})^q f(x,y), p,q = 0,1,2,... \quad (4.1.33)$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad (4.1.34)$$

and

$$\bar{y} = \frac{m_{01}}{m_{00}} \quad (4.1.35)$$

The central moments were computed using the image centroid, which is just similar to regular moments of any given image whose centre has been shifted to coincide with its centroid. Thus the central moments are invariant to image translation just like it's done with Zernike moments in section 4.1.3.1 of chapter 4.

The scale invariance was achieved as follows:

Let $f(x,y)$ represent any image (or image from the Flavia dataset); after scaling by a factor, say, α, we have $x' = \alpha x$, $y' = \alpha y$ and then :

$$m'_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x'^p y'^q f'(x',y')dx'dy' \quad (4.1.36)$$

$$m'_{pq} = \alpha^{p+q+2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^p y^q f(x,y)dxdy \quad (4.1.37)$$

$$m'_{pq} = \alpha^{p+q+2} m_{pq} \quad (4.1.38)$$

In similar manner to Equation 4.1.38, $\mu'_{pq} = \alpha^{p+q+2}\mu_{pq} \quad and \quad \mu'_{00} = \alpha^2\mu_{00}$

The normalized moments can now be computed as:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \ \gamma = \frac{p+q+2}{2}, \ p+q = 2,3,... \quad (4.1.39)$$

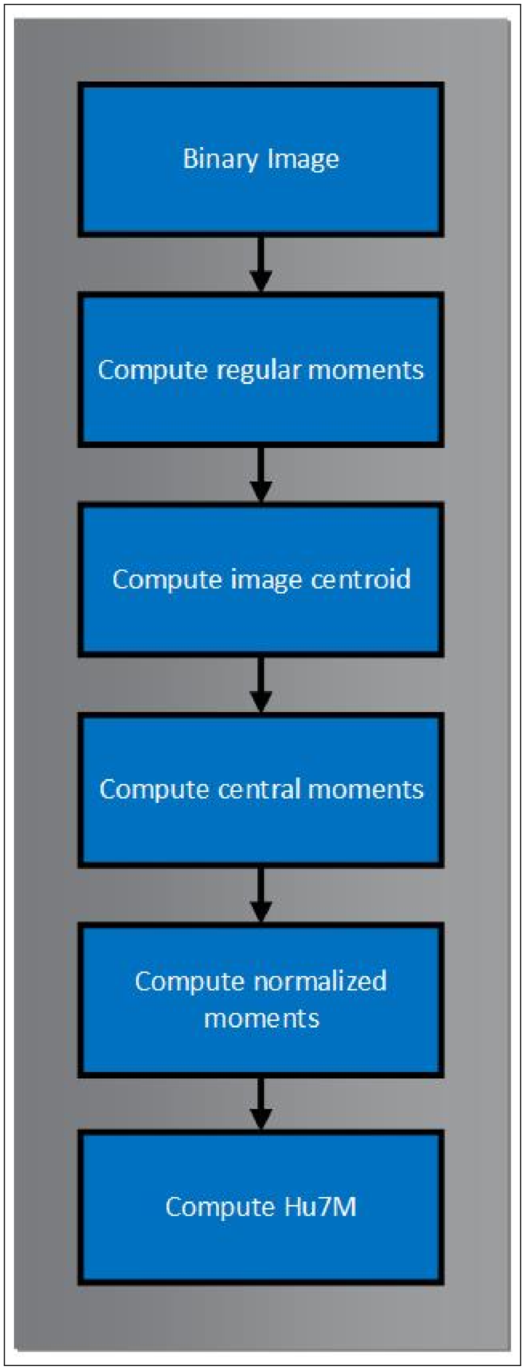


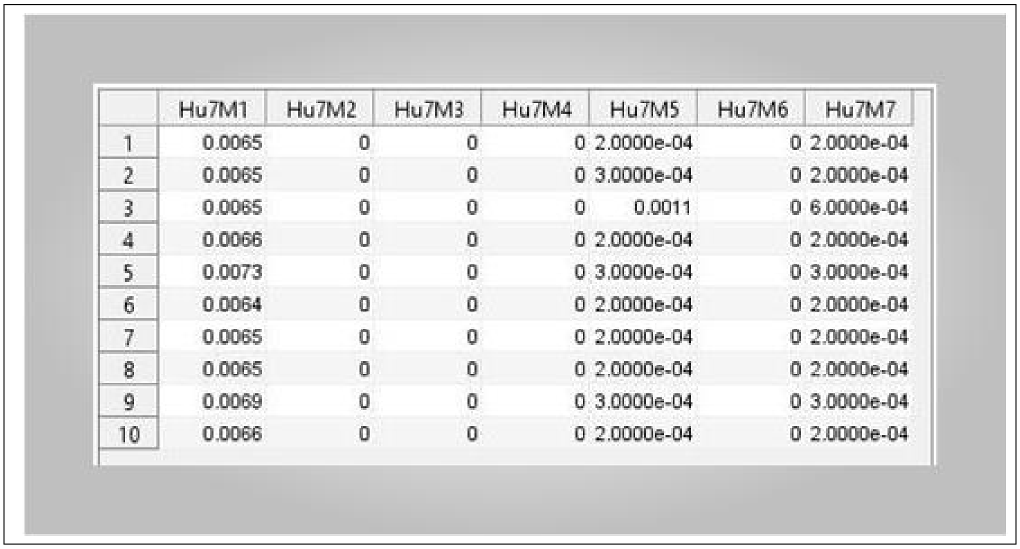
Figure 4.25: Computation of Hu’s 7 moments

The scale-invariance property of η_{pq} is easily established algebraically as :

$$\eta_{pq}^{'} = \frac{\eta_{pq}^{'}}{\eta_{00}^{'\gamma}} = \frac{\alpha^{p+q+2}\mu_{pq}}{\alpha^{2\gamma}\mu_{00}^{\gamma}} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{4.1.40}$$

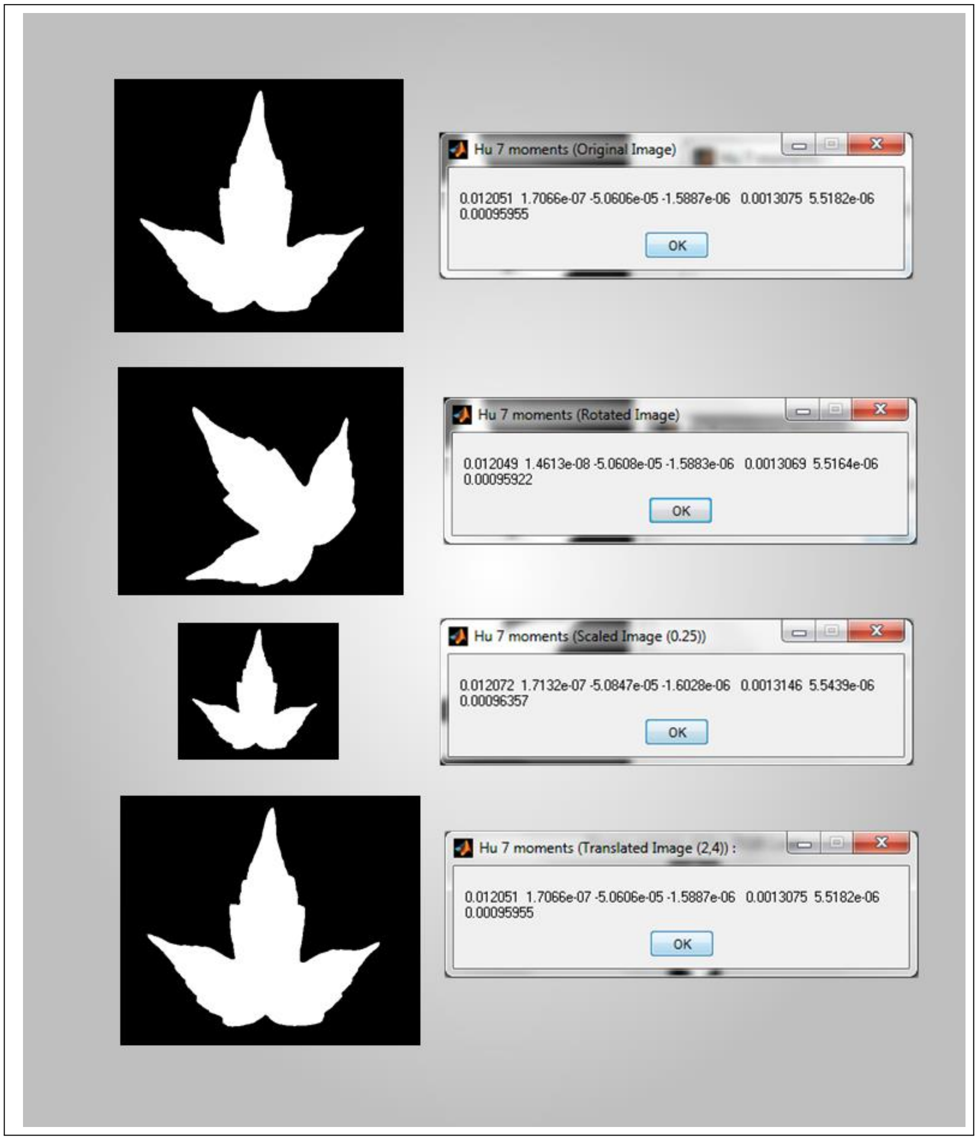
4.1.3.4 Results of feature extraction based on Hu7M

The Hu 7 moment extracted from the first 10 samples (the same species) from the Flavia dataset are listed below. Hu1 , Hu5, and Hu7 are more predictive than Hu2, Hu3, Hu4, and Hu6. The columns showing 0 as answers may display values greater than 0 if more precison or decimal display are needed. Figure 4.27 shows the invariant properties of Hu 7 moments with respect to scaling, rotation , and translation for a single image taken from the Flavia dataset.



	Hu7M1	Hu7M2	Hu7M3	Hu7M4	Hu7M5	Hu7M6	Hu7M7
1	0.0065	0	0	0	2.0000e-04	0	2.0000e-04
2	0.0065	0	0	0	3.0000e-04	0	2.0000e-04
3	0.0065	0	0	0	0.0011	0	6.0000e-04
4	0.0066	0	0	0	2.0000e-04	0	2.0000e-04
5	0.0073	0	0	0	3.0000e-04	0	3.0000e-04
6	0.0064	0	0	0	2.0000e-04	0	2.0000e-04
7	0.0065	0	0	0	2.0000e-04	0	2.0000e-04
8	0.0065	0	0	0	2.0000e-04	0	2.0000e-04
9	0.0069	0	0	0	3.0000e-04	0	3.0000e-04
10	0.0066	0	0	0	2.0000e-04	0	2.0000e-04

Figure 4.26: The Hu 7 moment extracted from the first 10 samples (the same species) from the Flavia dataset



Hu 7 moments (Original Image)
OK

```
0.012051  1.7066e-07 -5.0606e-05 -1.5887e-06   0.0013075  5.5182e-06
0.00095955
```

Hu 7 moments (Rotated Image)
OK

```
0.012049  1.4613e-08 -5.0608e-05 -1.5883e-06   0.0013069  5.5164e-06
0.00095922
```

Hu 7 moments (Scaled Image (0.25))
OK

```
0.012072  1.7132e-07 -5.0847e-05 -1.6028e-06   0.0013146  5.5439e-06
0.00096357
```

Hu 7 moments (Translated Image (2,4)) :
OK

```
0.012051  1.7066e-07 -5.0606e-05 -1.5887e-06   0.0013075  5.5182e-06
0.00095955
```

Figure 4.27: Affine properties of Hu 7 Moments for a single Flavia image

4.1.3.5 Legendre Moments (Shape features)

Another TRS-invariant descriptor used in this study is the Lengendre Moment which are based on the Legendre polynomial (Teh & Chin, 1988; Chong, Raveendran, & Mukundan, 2004; Hosny, 2007). The $nth-order$ Legendre polynomial is defined by

$$P_n(x) = \sum_{j=0}^{n} a_{nj}x^j = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n \tag{4.1.41}$$

or

$$P_n(x) = \frac{1}{2^n}\sum_{k=0}^{n/2}(-1)^k \frac{(2n-2k)!}{k!(n-k)!(n-2k)!}x^{n-2k} \tag{4.1.42}$$

The Legendre Polynomials have the generating function

$$\frac{1}{\sqrt{(1-2rx+r^2)}} = \sum_{s=0}^{\infty} r^s P_s(x), r < 1 \tag{4.1.43}$$

The recurrent formula of the Legendre polynomials is

$$P_{n+1}(x) = \frac{2n+1}{n+1}xP_n(x) - \frac{n}{n+1}P_{n-1}(x) \tag{4.1.44}$$

where $P_0(x) = 1, P_1(x) = x, \& p \in \mathbb{Z}^+$ such that $p > 1$

The Legendre polynomials in Equation 4.1.41 forms a complete orthogonal basis set on the interval [-1,1];

$$\int_{-1}^{+1} P_m(x)P_n(x)dx = \frac{2}{2m+1}\delta_{mn}, \tag{4.1.45}$$

where δ_{mn} is the Kronecker symbol. The $(m+n)th$ order of Legendre moment for a given image of intensity $f(x,y)$ defined on the square $[-1,1]\times[-1,1]$ is

$$L_{m,n} = \frac{(2m+1)(2n+1)}{4}\int_{-1}^{+1}\int_{-1}^{+1} P_m(x)P_n(y)f(x,y)dxdy \tag{4.1.46}$$

where $m,n = 0,1,2,....$

The images found in the Flavia dataset were all rectangular. The LM for digital images in square domain $N \times N$ is given as Equation 4.1.47.

$$L_{m,n} = \frac{(2m+1)(2n+1)}{(N-1)^2} \sum_{i=1}^{N} \sum_{j=1}^{N} P_m(x_i) P_n(y_j) f(x_i, y_j) dxdy \tag{4.1.47}$$

where

$$x_i = \frac{2i - N - 1}{N - 1} \tag{4.1.48}$$

and

$$y_j = \frac{2j - N - 1}{N - 1} \tag{4.1.49}$$

Algorithm 1 Computation of Legendre Moments

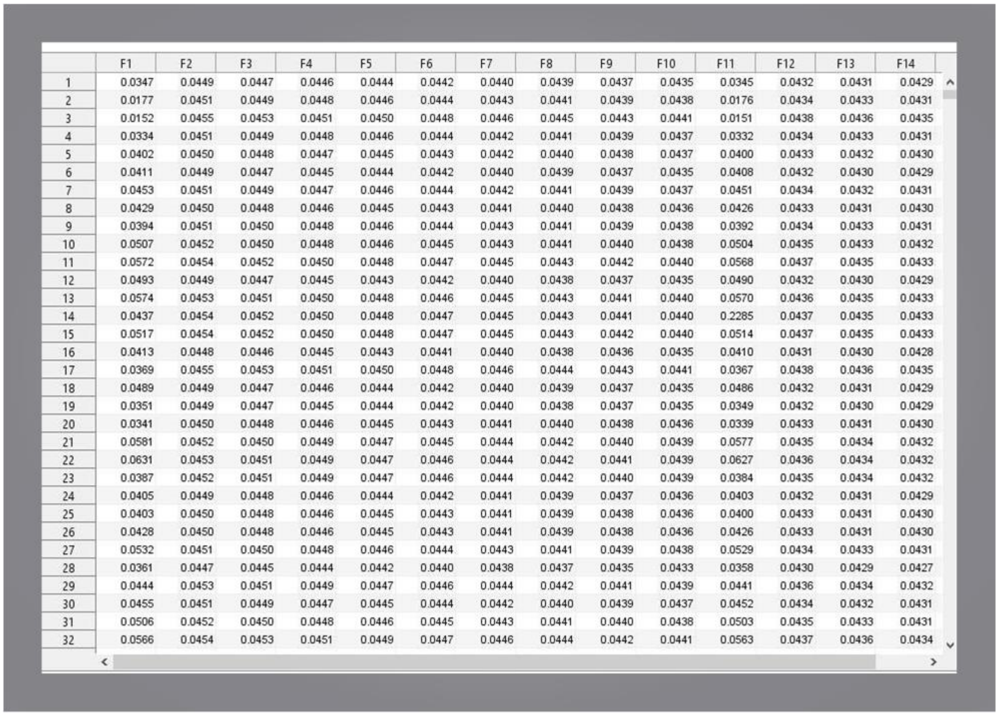
```
procedure  ComputeLegendreMoments()
    Input image $f(x_i, y_j)$, $,i = 1(1)M$, $,j = 1(1)N$
    Normalize $f(x,y)$
    Compute $\Delta x_i = \frac{2}{M}, \Delta y_j = \frac{2}{N}$
    for $i = 1(1)M$
            for $j = 1(1)N$
$x_i{}^* \mapsto -1 + (i - \frac{1}{2})\Delta x_i$
$y_j{}^* \mapsto -1 + (j - \frac{1}{2})\Delta y_j$
$L_{m,n} \mapsto \frac{(2m+1)(2n+1)}{4} \int_{-1}^{+1} \int_{-1}^{+1} P_m(x_i{}^*) P_n(y_j{}^*) f(x_i{}^*, y_j{}^*) dxdy$
    Endfor.
            Endfor
    Output $L_{m,n}$
end procedure
```

4.1.3.6 Results of feature extraction based on Legendre moment (LM)

The first 33 instances of 14 Legendre moment (LM or $L_{m,n}$) features extracted from the images of plant leaves in the Flavia dataset are shown in Figure 4.28. The numbers shown in this figure show that the LMs are very close in value. This is due to the fact that the LMs were computed from the images of plants from the same species. An affine transformation (or mapping) is being shown here using Algorithm 1 since the images in the database were stored in different scalings, translation and rotation. In other words the value of LM remains constant across different instances of the same images of plant species in the dataset.

The original images can be reconstructed back by using an infinite series expansion in terms of the Legendre polynomials over the square domain $[-1,1] \times [-1,1]$. This is expressed as Equation 4.1.50.

$$f(x,y) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} L_{m,n} P_m(x_i{}^*) P_n(y_j{}^*) \tag{4.1.50}$$



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
1	0.0347	0.0449	0.0447	0.0446	0.0444	0.0442	0.0440	0.0439	0.0437	0.0435	0.0345	0.0432	0.0431	0.0429
2	0.0177	0.0451	0.0449	0.0448	0.0446	0.0444	0.0443	0.0441	0.0439	0.0438	0.0176	0.0434	0.0433	0.0431
3	0.0152	0.0455	0.0453	0.0451	0.0450	0.0448	0.0446	0.0445	0.0443	0.0441	0.0151	0.0438	0.0436	0.0435
4	0.0334	0.0451	0.0449	0.0448	0.0446	0.0444	0.0442	0.0441	0.0439	0.0437	0.0332	0.0434	0.0433	0.0431
5	0.0402	0.0450	0.0448	0.0447	0.0445	0.0443	0.0442	0.0440	0.0438	0.0437	0.0400	0.0433	0.0432	0.0430
6	0.0411	0.0449	0.0447	0.0445	0.0444	0.0442	0.0440	0.0439	0.0437	0.0435	0.0408	0.0432	0.0430	0.0429
7	0.0453	0.0451	0.0449	0.0447	0.0446	0.0444	0.0442	0.0441	0.0439	0.0437	0.0451	0.0434	0.0432	0.0431
8	0.0429	0.0450	0.0448	0.0446	0.0445	0.0443	0.0441	0.0440	0.0438	0.0436	0.0426	0.0433	0.0431	0.0430
9	0.0394	0.0451	0.0450	0.0448	0.0446	0.0444	0.0443	0.0441	0.0439	0.0438	0.0392	0.0434	0.0433	0.0431
10	0.0507	0.0452	0.0450	0.0448	0.0446	0.0445	0.0443	0.0441	0.0440	0.0438	0.0504	0.0435	0.0433	0.0432
11	0.0572	0.0454	0.0452	0.0450	0.0448	0.0447	0.0445	0.0443	0.0442	0.0440	0.0568	0.0437	0.0435	0.0433
12	0.0493	0.0449	0.0447	0.0445	0.0443	0.0442	0.0440	0.0438	0.0437	0.0435	0.0490	0.0432	0.0430	0.0429
13	0.0574	0.0453	0.0451	0.0450	0.0448	0.0446	0.0445	0.0443	0.0441	0.0440	0.0570	0.0436	0.0435	0.0433
14	0.0437	0.0454	0.0452	0.0450	0.0448	0.0447	0.0445	0.0443	0.0441	0.0440	0.2285	0.0437	0.0435	0.0433
15	0.0517	0.0454	0.0452	0.0450	0.0448	0.0447	0.0445	0.0443	0.0442	0.0440	0.0514	0.0437	0.0435	0.0433
16	0.0413	0.0448	0.0446	0.0445	0.0443	0.0441	0.0440	0.0438	0.0436	0.0435	0.0410	0.0431	0.0430	0.0428
17	0.0369	0.0455	0.0453	0.0451	0.0450	0.0448	0.0446	0.0444	0.0443	0.0441	0.0367	0.0438	0.0436	0.0435
18	0.0489	0.0449	0.0447	0.0446	0.0444	0.0442	0.0440	0.0439	0.0437	0.0435	0.0486	0.0432	0.0431	0.0429
19	0.0351	0.0449	0.0447	0.0445	0.0444	0.0442	0.0440	0.0438	0.0437	0.0435	0.0349	0.0432	0.0430	0.0429
20	0.0341	0.0450	0.0448	0.0446	0.0445	0.0443	0.0441	0.0440	0.0438	0.0436	0.0339	0.0433	0.0431	0.0430
21	0.0581	0.0452	0.0450	0.0449	0.0447	0.0445	0.0444	0.0442	0.0440	0.0439	0.0577	0.0435	0.0434	0.0432
22	0.0631	0.0453	0.0451	0.0449	0.0447	0.0446	0.0444	0.0442	0.0441	0.0439	0.0627	0.0436	0.0434	0.0432
23	0.0387	0.0452	0.0451	0.0449	0.0447	0.0446	0.0444	0.0442	0.0440	0.0439	0.0384	0.0435	0.0434	0.0432
24	0.0405	0.0449	0.0448	0.0446	0.0444	0.0442	0.0441	0.0439	0.0437	0.0436	0.0403	0.0432	0.0431	0.0429
25	0.0403	0.0450	0.0448	0.0446	0.0445	0.0443	0.0441	0.0439	0.0438	0.0436	0.0400	0.0433	0.0431	0.0430
26	0.0428	0.0450	0.0448	0.0446	0.0445	0.0443	0.0441	0.0439	0.0438	0.0436	0.0426	0.0433	0.0431	0.0430
27	0.0532	0.0451	0.0450	0.0448	0.0446	0.0444	0.0443	0.0441	0.0439	0.0438	0.0529	0.0434	0.0433	0.0431
28	0.0361	0.0447	0.0445	0.0444	0.0442	0.0440	0.0438	0.0437	0.0435	0.0433	0.0358	0.0430	0.0429	0.0427
29	0.0444	0.0453	0.0451	0.0449	0.0447	0.0446	0.0444	0.0442	0.0441	0.0439	0.0441	0.0436	0.0434	0.0432
30	0.0455	0.0451	0.0449	0.0447	0.0445	0.0444	0.0442	0.0440	0.0439	0.0437	0.0452	0.0434	0.0432	0.0431
31	0.0506	0.0452	0.0450	0.0448	0.0446	0.0445	0.0443	0.0441	0.0440	0.0438	0.0503	0.0435	0.0433	0.0431
32	0.0566	0.0454	0.0453	0.0451	0.0449	0.0447	0.0446	0.0444	0.0442	0.0441	0.0563	0.0437	0.0436	0.0434

Figure 4.28: The first 33 instances of 14 Legendre moment features extracted from the images of plant leaves in the Flavia dataset

4.1.3.7 Texture Features

The texture of an image refers to the local variation in brightness from one pixel point to the next or within a small region. It is also a measure of surface roughness (Russ, 2011) The texture properties (Contrast, Correlation, Energy, and Homogeneity) and their variation such as maximum probability, sum of squares of variance and sum of average were also used in this work. Textural features are often needed as they provide additional botanical information such as leaf venation, leaf pubescence, leaf lesion and insect damage (George, 2011). For each pixel in the image of the leaf, texture features help in determining the histogram of gray levels in predefined neighbouring region centred on that pixel (Kpalma & Ronsin, 2007). A Gray level co-occurrence matrix (GLCM) which is related to the texture features is a tabular topology of the frequency of occurrence of different combinations of intensity information (grey levels) in an image. In other words, the GLCM examines how often a pixel with gray-level (grayscale intensity) value i occurs horizontally adjacent to another pixel with the value j. The GLCM, according to Haralick, Shanmugam, and Dinstein (1973), can be defined by

$$GLCM_{\delta x,\delta y}(i,j)=\sum_{p=1}^{n}\sum_{q=1}^{m}\begin{cases}1, & \text{if } I(p,q)=i \text{ and } I(p+\delta x,q+\delta y)=j\\ 0, & \text{otherwise}\end{cases} \tag{4.1.51}$$

where matrix $GLCM \in I(n,m)$ is parameterized by an offset $(\delta x,\delta y)$ and $I(n,m)$ is the digital image. $GLCM$ has been used by some researchers developing recognition systems to capture texture features as reported in (Dobrescu, Dobrescu, Mocanu, & Popescu, 2010) and (Kekre, Thepade, & Sarode, 2010; Jiang, Wang, & Zhang, 2008). The texture of an image is derived from the spatial variation of its pixel intensities (Tuceryan & Jain, 1998). The $GLCM$ was used to capture the spatial dependence of gray-level intensities for different angles (0°, 45°, 90°, and 135°) of pixel relativity for plant's leaf recognition. A probability-density function (pdf) was then applied on each matrix to calculate different textural parameters. If $p(x,y)$ is the gray-level value at the coordinate (x,y) for any image from the database, the following texture properties are defined accordingly:

1. Entropy: This is given as :

$$Entropy = H = \sum_x \sum_y p(x,y) log p(x,y) \tag{4.1.52}$$

   The entropy measures the randomness in the texture of the image.

2. Inverse difference:

$$InvDiff = \sum_x \sum_y \frac{1}{1+(x-y)^2} p(x,y) \tag{4.1.53}$$

3. Energy:

$$Energy = \sum_x \sum_y p(x,y)^2 \tag{4.1.54}$$

4. Contrast:

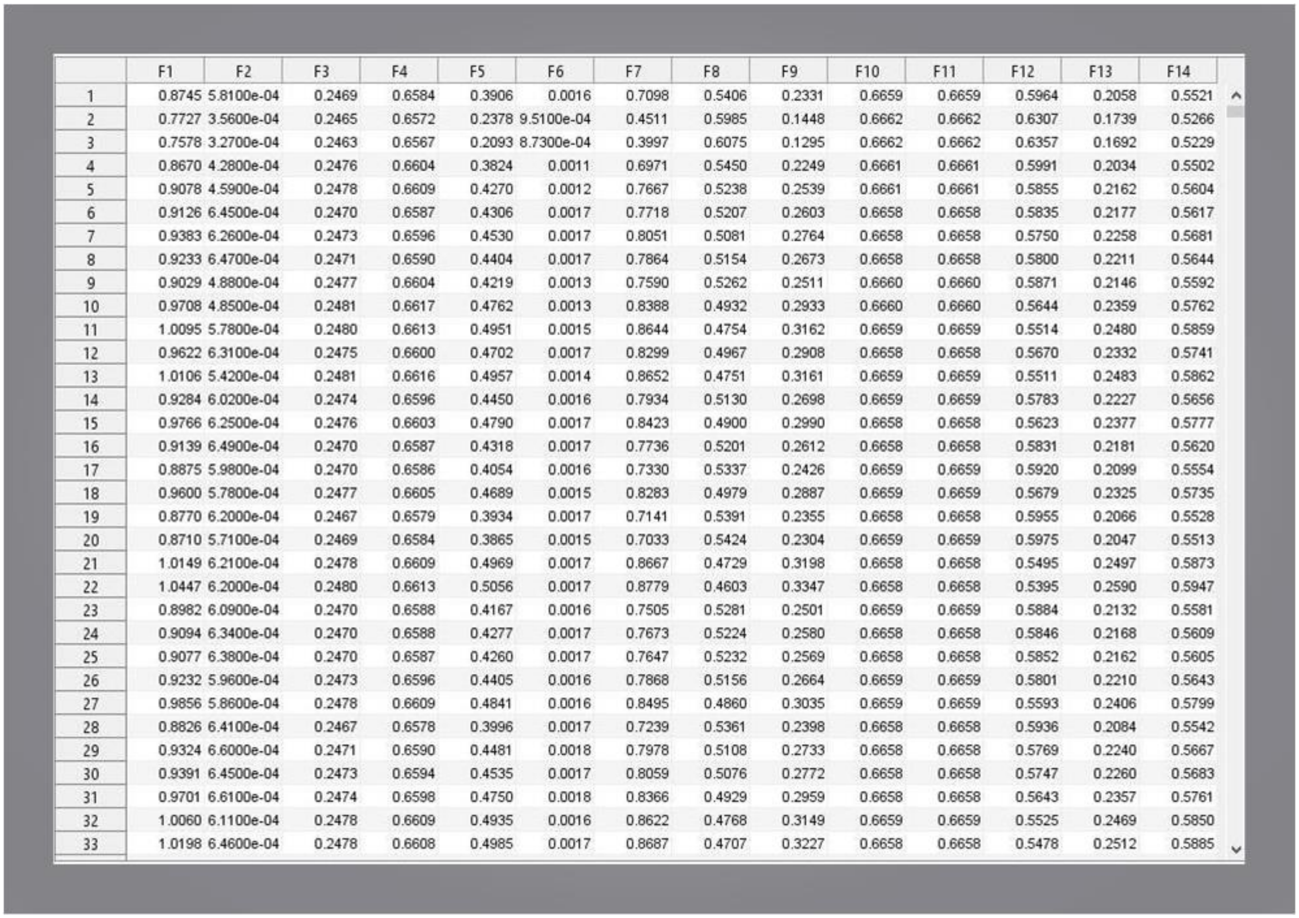
$$Contrast = \sum_x \sum_y (x-y)^2 p(x,y) \tag{4.1.55}$$

4.1.3.8 Results of feature extraction based on texture properties

The texture features used in this work are computed based on the formular 4.1.51 and are indexed by the vector $v5$ = [48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]. Thus the texture features can be selected from the original feature space by using the following syntax:

PhDDataSetTextureFeatures = PhDDataSet (:, [48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]).

Part of the features associated with variable PhDDataSetTextureFeatures are shown in Figure 4.29.



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
1	0.8745	5.8100e-04	0.2469	0.6584	0.3906	0.0016	0.7098	0.5406	0.2331	0.6659	0.6659	0.5964	0.2058	0.5521
2	0.7727	3.5600e-04	0.2465	0.6572	0.2378	9.5100e-04	0.4511	0.5985	0.1448	0.6662	0.6662	0.6307	0.1739	0.5266
3	0.7578	3.2700e-04	0.2463	0.6567	0.2093	8.7300e-04	0.3997	0.6075	0.1295	0.6662	0.6662	0.6357	0.1692	0.5229
4	0.8670	4.2800e-04	0.2476	0.6604	0.3824	0.0011	0.6971	0.5450	0.2249	0.6661	0.6661	0.5991	0.2034	0.5502
5	0.9078	4.5900e-04	0.2478	0.6609	0.4270	0.0012	0.7667	0.5238	0.2539	0.6661	0.6661	0.5855	0.2162	0.5604
6	0.9126	6.4500e-04	0.2470	0.6587	0.4306	0.0017	0.7718	0.5207	0.2603	0.6658	0.6658	0.5835	0.2177	0.5617
7	0.9383	6.2600e-04	0.2473	0.6596	0.4530	0.0017	0.8051	0.5081	0.2764	0.6658	0.6658	0.5750	0.2258	0.5681
8	0.9233	6.4700e-04	0.2471	0.6590	0.4404	0.0017	0.7864	0.5154	0.2673	0.6658	0.6658	0.5800	0.2211	0.5644
9	0.9029	4.8800e-04	0.2477	0.6604	0.4219	0.0013	0.7590	0.5262	0.2511	0.6660	0.6660	0.5871	0.2146	0.5592
10	0.9708	4.8500e-04	0.2481	0.6617	0.4762	0.0013	0.8388	0.4932	0.2933	0.6660	0.6660	0.5644	0.2359	0.5762
11	1.0095	5.7800e-04	0.2480	0.6613	0.4951	0.0015	0.8644	0.4754	0.3162	0.6659	0.6659	0.5514	0.2480	0.5859
12	0.9622	6.3100e-04	0.2475	0.6600	0.4702	0.0017	0.8299	0.4967	0.2908	0.6658	0.6658	0.5670	0.2332	0.5741
13	1.0106	5.4200e-04	0.2481	0.6616	0.4957	0.0014	0.8652	0.4751	0.3161	0.6659	0.6659	0.5511	0.2483	0.5862
14	0.9284	6.0200e-04	0.2474	0.6596	0.4450	0.0016	0.7934	0.5130	0.2698	0.6659	0.6659	0.5783	0.2227	0.5656
15	0.9766	6.2500e-04	0.2476	0.6603	0.4790	0.0017	0.8423	0.4900	0.2990	0.6658	0.6658	0.5623	0.2377	0.5777
16	0.9139	6.4900e-04	0.2470	0.6587	0.4318	0.0017	0.7736	0.5201	0.2612	0.6658	0.6658	0.5831	0.2181	0.5620
17	0.8875	5.9800e-04	0.2470	0.6586	0.4054	0.0016	0.7330	0.5337	0.2426	0.6659	0.6659	0.5920	0.2099	0.5554
18	0.9600	5.7800e-04	0.2477	0.6605	0.4689	0.0015	0.8283	0.4979	0.2887	0.6659	0.6659	0.5679	0.2325	0.5735
19	0.8770	6.2000e-04	0.2467	0.6579	0.3934	0.0017	0.7141	0.5391	0.2355	0.6658	0.6658	0.5955	0.2066	0.5528
20	0.8710	5.7100e-04	0.2469	0.6584	0.3865	0.0015	0.7033	0.5424	0.2304	0.6659	0.6659	0.5975	0.2047	0.5513
21	1.0149	6.2100e-04	0.2478	0.6609	0.4969	0.0017	0.8667	0.4729	0.3198	0.6658	0.6658	0.5495	0.2497	0.5873
22	1.0447	6.2000e-04	0.2480	0.6613	0.5056	0.0017	0.8779	0.4603	0.3347	0.6658	0.6658	0.5395	0.2590	0.5947
23	0.8982	6.0900e-04	0.2470	0.6588	0.4167	0.0016	0.7505	0.5281	0.2501	0.6659	0.6659	0.5884	0.2132	0.5581
24	0.9094	6.3400e-04	0.2470	0.6588	0.4277	0.0017	0.7673	0.5224	0.2580	0.6658	0.6658	0.5846	0.2168	0.5609
25	0.9077	6.3800e-04	0.2470	0.6587	0.4260	0.0017	0.7647	0.5232	0.2569	0.6658	0.6658	0.5852	0.2162	0.5605
26	0.9232	5.9600e-04	0.2473	0.6596	0.4405	0.0016	0.7868	0.5156	0.2664	0.6659	0.6659	0.5801	0.2210	0.5643
27	0.9856	5.8600e-04	0.2478	0.6609	0.4841	0.0016	0.8495	0.4860	0.3035	0.6659	0.6659	0.5593	0.2406	0.5799
28	0.8826	6.4100e-04	0.2467	0.6578	0.3996	0.0017	0.7239	0.5361	0.2398	0.6658	0.6658	0.5936	0.2084	0.5542
29	0.9324	6.6000e-04	0.2471	0.6590	0.4481	0.0018	0.7978	0.5108	0.2733	0.6658	0.6658	0.5769	0.2240	0.5667
30	0.9391	6.4500e-04	0.2473	0.6594	0.4535	0.0017	0.8059	0.5076	0.2772	0.6658	0.6658	0.5747	0.2260	0.5683
31	0.9701	6.6100e-04	0.2474	0.6598	0.4750	0.0018	0.8366	0.4929	0.2959	0.6658	0.6658	0.5643	0.2357	0.5761
32	1.0060	6.1100e-04	0.2478	0.6609	0.4935	0.0016	0.8622	0.4768	0.3149	0.6659	0.6659	0.5525	0.2469	0.5850
33	1.0198	6.4600e-04	0.2478	0.6608	0.4985	0.0017	0.8687	0.4707	0.3227	0.6658	0.6658	0.5478	0.2512	0.5885

Figure 4.29: The first 33 instances of 14 texture features extracted from the images of plant leaves in the Flavia dataset

4.1.3.9 Geometric and Morphological features (Shape features)

The geometric and morphological features extracted from the Flavia dataset are defined in the following section.

1. Physiological Length: This is the distance between the two terminals of a leaf (apex and stalk point).

2. Physiological Width: This is the perpendicular distance across the physiological length of a leaf.

3. Leaf Area: This is the total number of pixels that constitute an image. From the regular moment, we have :

   $$m_{pq} = \int_x \int_y x^p y^q f(x,y)dxdy \tag{4.1.56}$$

   and central moment as:

   $$\mu_{pq} = \int_x \int_y (x-x_c)^p (y-y_c)^q f(x,y)dxdy \tag{4.1.57}$$

   The area (spatial moment of zero order) of binary object is thus defined as

   $$A = m_{0,0} \tag{4.1.58}$$

4. Aspect Ratio(A.R): This is also called eccentricity and is defined as ratio between length of the leaf minor axis and the length of the leaf major axis (Abdul et al., 2012). The eccentricity can also be defined as

   $$\varepsilon = \frac{(\mu_{2,0}-\mu_{0,2})^2 - 4\mu_{1,1}^2}{(\mu_{2,0}+\mu_{0,2})^2} \tag{4.1.59}$$

   For $\varepsilon = 0$, the image or object beind described is a circle while for $\varepsilon = 1$, the object is a line.

5. Circularity: This is a measure of similarity between a 2D shape is and a circle. It is the ratio between area of the leaf and the square of its perimeter (Russ, 2011).

6. Solidity: This is defined as the ratio between the area of the leaf and the area of its convex hull (Russ, 2011).

7. Convexity: This is the ratio between the convex hull perimeter of the leaf and the perimeter of the leaf (Russ, 2011).

8. Form Factor: This feature describes the difference between a leaf and a circle, where is the leaf area and is the perimeter of the leaf (Wu et al., 2007).

9. Narrow Factor: This is the ratio of the diameter and length of the leaf. (Wu et al., 2007).

10. Form Factor: This feature describes the difference between a leaf and a circle.

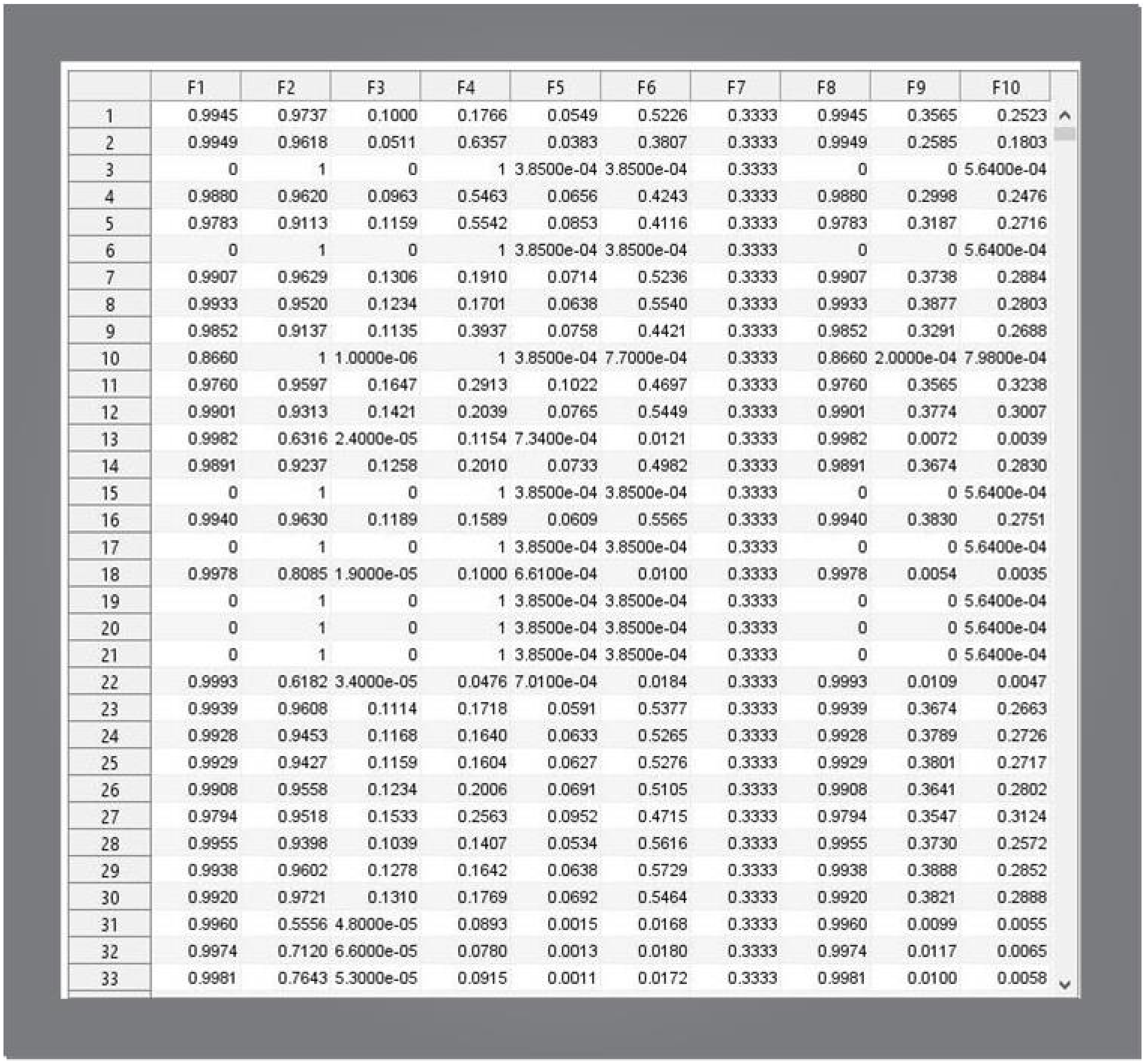
4.1.3.10 Results of feature extraction based on Geometry and Morphology

The geometric features defined here have been computed and are indexed by the vector $v6$ = [70, 71, 72, 73, 74, 75, 76, 77, 78, 79] in the feature space. These features (some shown in Figure 4.30) can be selected from the original feature space by using the syntax:

PhDDataSetGeometricFeatures = PhDDataSet (:, [70, 71, 72, 73, 74, 75, 76, 77, 78, 79]).

4.1.3.11 Colour features

Colour features are associated with colour distribution inside the image . Colour moments can be extracted from component on the leaf by the computation of statistical measures such as mean,



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
1	0.9945	0.9737	0.1000	0.1766	0.0549	0.5226	0.3333	0.9945	0.3565	0.2523
2	0.9949	0.9618	0.0511	0.6357	0.0383	0.3807	0.3333	0.9949	0.2585	0.1803
3	0	1	0	1	3.8500e-04	3.8500e-04	0.3333	0	0	5.6400e-04
4	0.9880	0.9620	0.0963	0.5463	0.0656	0.4243	0.3333	0.9880	0.2998	0.2476
5	0.9783	0.9113	0.1159	0.5542	0.0853	0.4116	0.3333	0.9783	0.3187	0.2716
6	0	1	0	1	3.8500e-04	3.8500e-04	0.3333	0	0	5.6400e-04
7	0.9907	0.9629	0.1306	0.1910	0.0714	0.5236	0.3333	0.9907	0.3738	0.2884
8	0.9933	0.9520	0.1234	0.1701	0.0638	0.5540	0.3333	0.9933	0.3877	0.2803
9	0.9852	0.9137	0.1135	0.3937	0.0758	0.4421	0.3333	0.9852	0.3291	0.2688
10	0.8660	1	1.0000e-06	1	3.8500e-04	7.7000e-04	0.3333	0.8660	2.0000e-04	7.9800e-04
11	0.9760	0.9597	0.1647	0.2913	0.1022	0.4697	0.3333	0.9760	0.3565	0.3238
12	0.9901	0.9313	0.1421	0.2039	0.0765	0.5449	0.3333	0.9901	0.3774	0.3007
13	0.9982	0.6316	2.4000e-05	0.1154	7.3400e-04	0.0121	0.3333	0.9982	0.0072	0.0039
14	0.9891	0.9237	0.1258	0.2010	0.0733	0.4982	0.3333	0.9891	0.3674	0.2830
15	0	1	0	1	3.8500e-04	3.8500e-04	0.3333	0	0	5.6400e-04
16	0.9940	0.9630	0.1189	0.1589	0.0609	0.5565	0.3333	0.9940	0.3830	0.2751
17	0	1	0	1	3.8500e-04	3.8500e-04	0.3333	0	0	5.6400e-04
18	0.9978	0.8085	1.9000e-05	0.1000	6.6100e-04	0.0100	0.3333	0.9978	0.0054	0.0035
19	0	1	0	1	3.8500e-04	3.8500e-04	0.3333	0	0	5.6400e-04
20	0	1	0	1	3.8500e-04	3.8500e-04	0.3333	0	0	5.6400e-04
21	0	1	0	1	3.8500e-04	3.8500e-04	0.3333	0	0	5.6400e-04
22	0.9993	0.6182	3.4000e-05	0.0476	7.0100e-04	0.0184	0.3333	0.9993	0.0109	0.0047
23	0.9939	0.9608	0.1114	0.1718	0.0591	0.5377	0.3333	0.9939	0.3674	0.2663
24	0.9928	0.9453	0.1168	0.1640	0.0633	0.5265	0.3333	0.9928	0.3789	0.2726
25	0.9929	0.9427	0.1159	0.1604	0.0627	0.5276	0.3333	0.9929	0.3801	0.2717
26	0.9908	0.9558	0.1234	0.2006	0.0691	0.5105	0.3333	0.9908	0.3641	0.2802
27	0.9794	0.9518	0.1533	0.2563	0.0952	0.4715	0.3333	0.9794	0.3547	0.3124
28	0.9955	0.9398	0.1039	0.1407	0.0534	0.5616	0.3333	0.9955	0.3730	0.2572
29	0.9938	0.9602	0.1278	0.1642	0.0638	0.5729	0.3333	0.9938	0.3888	0.2852
30	0.9920	0.9721	0.1310	0.1769	0.0692	0.5464	0.3333	0.9920	0.3821	0.2888
31	0.9960	0.5556	4.8000e-05	0.0893	0.0015	0.0168	0.3333	0.9960	0.0099	0.0055
32	0.9974	0.7120	6.6000e-05	0.0780	0.0013	0.0180	0.3333	0.9974	0.0117	0.0065
33	0.9981	0.7643	5.3000e-05	0.0915	0.0011	0.0172	0.3333	0.9981	0.0100	0.0058

Figure 4.30: The first 33 instances of all geometric features extracted

standard deviation, skewness, and kurtosis (Martinez & Martinez., 2002). These features can be calculated using the formulas in (Equations 4.1.60 to 4.1.63)

$$\mu = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} P_{ij} \tag{4.1.60}$$

$$\sigma = \sqrt{\frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (P_{ij} - \mu)^2} \tag{4.1.61}$$

$$\theta = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} (P_{ij} - \mu)^3}{MN\sigma^3} \tag{4.1.62}$$

$$\gamma = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} (P_{ij} - \mu)^4}{MN\sigma^4} - 3 \tag{4.1.63}$$

where M × N is size of the image , P_{ij} is the value of the colour on row i_{th} and column j_{th}. The HSI (Hue, Saturation, Intensity) color space is the ideal system for developing image-processing algorithms based on color descriptions that are natural to human.(R. C. Gonzalez et al., 2009). The images in the Flavia dataset were first converted from rgb color image to hsi color image. The conversion from RGB system to HSI is given as follows:

$$H = \begin{cases} \theta & \text{if} \quad B \leq G \\ 360 - \theta & \text{if} \quad B > G \end{cases} \tag{4.1.64}$$

where

$$\theta = \cos^{-1}\left[\frac{0.5\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{0.5}}\right] \tag{4.1.65}$$

The saturation is given by

$$S = 1 - \left[\frac{3}{(R+G+B)}\right]\left[min(R,G,B)\right] \tag{4.1.66}$$

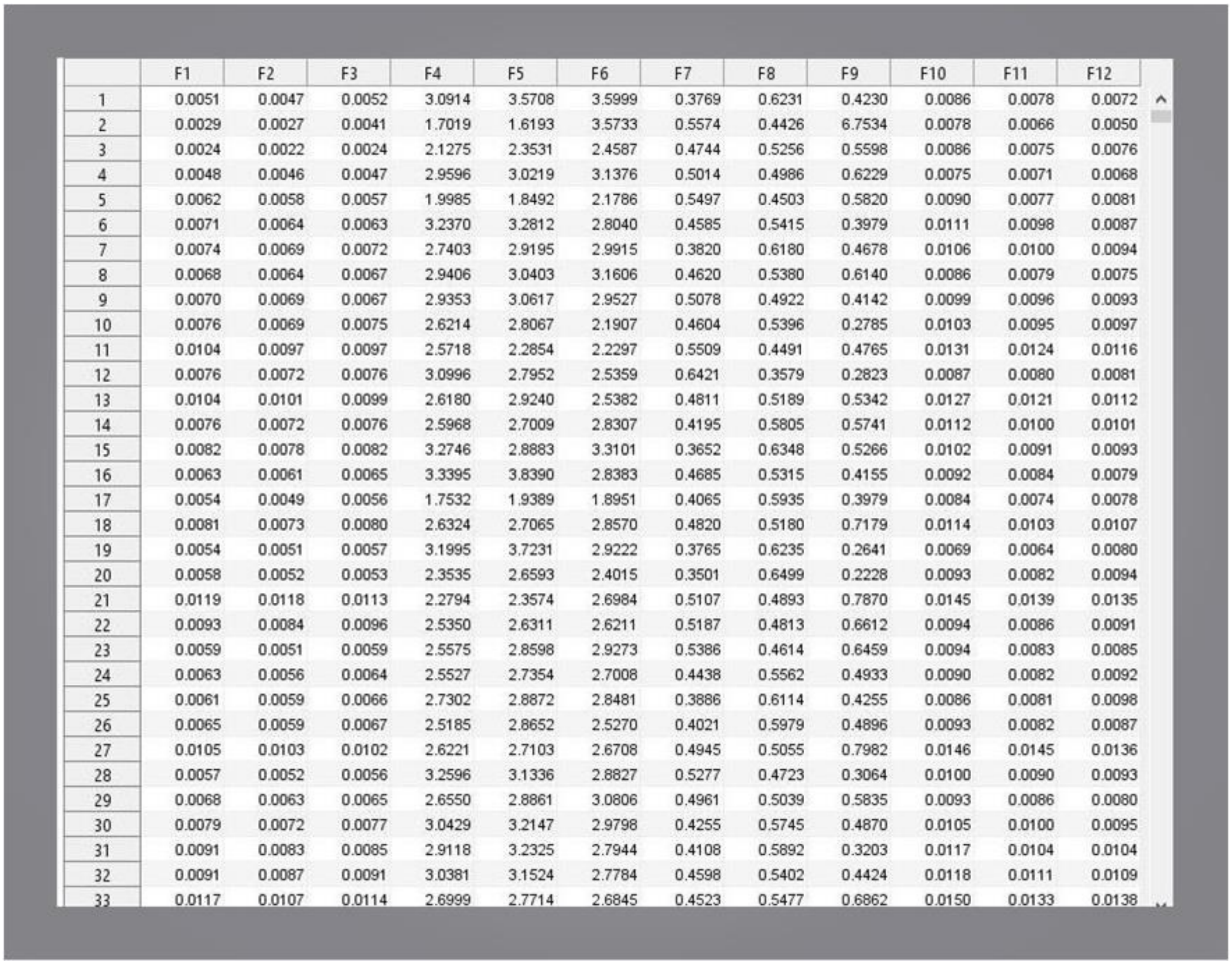
while the Intensity is given by

$$I = \frac{1}{3}(R+G+B) \tag{4.1.67}$$

In this study, the colour features based on formulas in Equations 4.1.60, 4.1.61, 4.1.62 & 4.1.63 and were extracted from the Flavia dataset as shown in Figure 4.31.

4.1.3.12 Results of feature extraction based on colour

A sample (first 33) color features generated from the Flavia dataset (based on the formulas in (Equations 4.1.60 to 4.1.63)) are given in Figure 4.31.



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
1	0.0051	0.0047	0.0052	3.0914	3.5708	3.5999	0.3769	0.6231	0.4230	0.0086	0.0078	0.0072
2	0.0029	0.0027	0.0041	1.7019	1.6193	3.5733	0.5574	0.4426	6.7534	0.0078	0.0066	0.0050
3	0.0024	0.0022	0.0024	2.1275	2.3531	2.4587	0.4744	0.5256	0.5598	0.0086	0.0075	0.0076
4	0.0048	0.0046	0.0047	2.9596	3.0219	3.1376	0.5014	0.4986	0.6229	0.0075	0.0071	0.0068
5	0.0062	0.0058	0.0057	1.9985	1.8492	2.1786	0.5497	0.4503	0.5820	0.0090	0.0077	0.0081
6	0.0071	0.0064	0.0063	3.2370	3.2812	2.8040	0.4585	0.5415	0.3979	0.0111	0.0098	0.0087
7	0.0074	0.0069	0.0072	2.7403	2.9195	2.9915	0.3820	0.6180	0.4678	0.0106	0.0100	0.0094
8	0.0068	0.0064	0.0067	2.9406	3.0403	3.1606	0.4620	0.5380	0.6140	0.0086	0.0079	0.0075
9	0.0070	0.0069	0.0067	2.9353	3.0617	2.9527	0.5078	0.4922	0.4142	0.0099	0.0096	0.0093
10	0.0076	0.0069	0.0075	2.6214	2.8067	2.1907	0.4604	0.5396	0.2785	0.0103	0.0095	0.0097
11	0.0104	0.0097	0.0097	2.5718	2.2854	2.2297	0.5509	0.4491	0.4765	0.0131	0.0124	0.0116
12	0.0076	0.0072	0.0076	3.0996	2.7952	2.5359	0.6421	0.3579	0.2823	0.0087	0.0080	0.0081
13	0.0104	0.0101	0.0099	2.6180	2.9240	2.5382	0.4811	0.5189	0.5342	0.0127	0.0121	0.0112
14	0.0076	0.0072	0.0076	2.5968	2.7009	2.8307	0.4195	0.5805	0.5741	0.0112	0.0100	0.0101
15	0.0082	0.0078	0.0082	3.2746	2.8883	3.3101	0.3652	0.6348	0.5266	0.0102	0.0091	0.0093
16	0.0063	0.0061	0.0065	3.3395	3.8390	2.8383	0.4685	0.5315	0.4155	0.0092	0.0084	0.0079
17	0.0054	0.0049	0.0056	1.7532	1.9389	1.8951	0.4065	0.5935	0.3979	0.0084	0.0074	0.0078
18	0.0081	0.0073	0.0080	2.6324	2.7065	2.8570	0.4820	0.5180	0.7179	0.0114	0.0103	0.0107
19	0.0054	0.0051	0.0057	3.1995	3.7231	2.9222	0.3765	0.6235	0.2641	0.0069	0.0064	0.0080
20	0.0058	0.0052	0.0053	2.3535	2.6593	2.4015	0.3501	0.6499	0.2228	0.0093	0.0082	0.0094
21	0.0119	0.0118	0.0113	2.2794	2.3574	2.6984	0.5107	0.4893	0.7870	0.0145	0.0139	0.0135
22	0.0093	0.0084	0.0096	2.5350	2.6311	2.6211	0.5187	0.4813	0.6612	0.0094	0.0086	0.0091
23	0.0059	0.0051	0.0059	2.5575	2.8598	2.9273	0.5386	0.4614	0.6459	0.0094	0.0083	0.0085
24	0.0063	0.0056	0.0064	2.5527	2.7354	2.7008	0.4438	0.5562	0.4933	0.0090	0.0082	0.0092
25	0.0061	0.0059	0.0066	2.7302	2.8872	2.8481	0.3886	0.6114	0.4255	0.0086	0.0081	0.0098
26	0.0065	0.0059	0.0067	2.5185	2.8652	2.5270	0.4021	0.5979	0.4896	0.0093	0.0082	0.0087
27	0.0105	0.0103	0.0102	2.6221	2.7103	2.6708	0.4945	0.5055	0.7982	0.0146	0.0145	0.0136
28	0.0057	0.0052	0.0056	3.2596	3.1336	2.8827	0.5277	0.4723	0.3064	0.0100	0.0090	0.0093
29	0.0068	0.0063	0.0065	2.6550	2.8861	3.0806	0.4961	0.5039	0.5835	0.0093	0.0086	0.0080
30	0.0079	0.0072	0.0077	3.0429	3.2147	2.9798	0.4255	0.5745	0.4870	0.0105	0.0100	0.0095
31	0.0091	0.0083	0.0085	2.9118	3.2325	2.7944	0.4108	0.5892	0.3203	0.0117	0.0104	0.0104
32	0.0091	0.0087	0.0091	3.0381	3.1524	2.7784	0.4598	0.5402	0.4424	0.0118	0.0111	0.0109
33	0.0117	0.0107	0.0114	2.6999	2.7714	2.6845	0.4523	0.5477	0.6862	0.0150	0.0133	0.0138

Figure 4.31: The first 33 instances of all colour features extracted the images of leaves in the Flavia dataset

4.2 Summary of Feature Set

The complete dataset for this work comprises of ZMs, FDs, Lengendre Moments, Hu 7 Moments, Texture, Geometrical properties and Color features. The variables $F_i, i = 1(1)N_f$, in Table 4.3 and Table 4.4 represent the features needed for this work. The N_f is the total number of image descriptors (features) in each table. Thus the feature space of this work is a $R^{r \times m}$ matrix, where r, (number of observations)$= \{1907, 1600\}$ and m, (number of attributes or futures required)$= 100$ for non-color features and 112 for both color-based + non-color respectively. The rational for using two different dataset is to validate the techniques used in this work beyond so that results across different dataset can be compared. The complete feature set from this chapter will be further analysed using GA, PSO, and PCA in Chapter 5 & 6 respectively.

Table 4.3: PhD Dataset 1 (Non-colour features)

SN	Descriptor	Feature Index	No of features
1	Zernike Moments	$F_1, F_2, F_3, ...F_{20}$	20
2	Lengendre Moments	$F_{21}, F_{22}, F_{23}, ...F_{40}$	20
3	Hu 7 Moments	$F_{41}, F_{42}, F_{43}, ...F_{47}$	7
4	Texture Features	$F_{48}, F_{49}, F_{50}, ...F_{69}$	22
5	Geometric Features	$F_{70}, F_{71}, F_{70}, ..., F_{79}$	10
6	Fourier Descriptors	$F_{80}, F_{81}, F_{82}, ..., F_{100}$	21

Table 4.4: PhD Dataset 2 (both color & non-colour features)

SN	Descriptor	Feature Index	No of features
1	Zernike Moments	$F_1, F_2, F_3, ...F_{20}$	20
2	Lengendre Moments	$F_{21}, F_{22}, F_{23}, ...F_{40}$	20
3	Hu 7 Moments	$F_{41}, F_{42}, F_{43}, ...F_{47}$	7
4	Texture Features	$F_{48}, F_{49}, F_{50}, ...F_{69}$	22
5	Geometric Features	$F_{70}, F_{71}, F_{70}, ..., F_{79}$	10
6	Fourier Descriptors	$F_{80}, F_{81}, F_{82}, ..., F_{100}$	21
7	Colour features	$F_{101}, F_{102}, F_{103}, ..., F_{112}$	12

Chapter 5

A Genetic Algorithm-Based Feature Selection

5.1 Introduction

In this chapter, a genetic algorithm -based feature selection is provided using the features generated from Chapter 4. As documented in (O. Babatunde, Armstrong, Leng, & Diepeveen, 2015a), some of the available feature selection techniques include Principal Component Analysis (PCA), Particle Swarm Optimization (PSO), Genetic Algorithm (GA). As such, a PSO, PCA, and GA-based feature selection (a subspace or manifold projection technique) will be used to reduce the number of features needed by the PNN Classifier in this work. Feature analysis on the other hand, is about the examination of extracted features from the given set of images and determining the kind of functional relationships that mighy exist between the selected features and seeing how they can be used to solve the imaging problem under consideration. Generally speaking, feature selection is a discrimination between two finite-point sets in the original n-dimensional feature space $\boldsymbol{R^n}$ by a separating hyperplane that utilizes as few of the features as possible. The findings from this chapter has been published in the paper (O. Babatunde et al., 2014b).

Definition 5.1.1. A Feature Subset Selection (FSS) is an operator $\boldsymbol{Fs}$ or a map from an m-dimensional feature space (input space) to n-dimensional feature space (output) given in mapping,

$$F_s : R^{r\times m} \mapsto R^{r\times n} \tag{5.1.1}$$

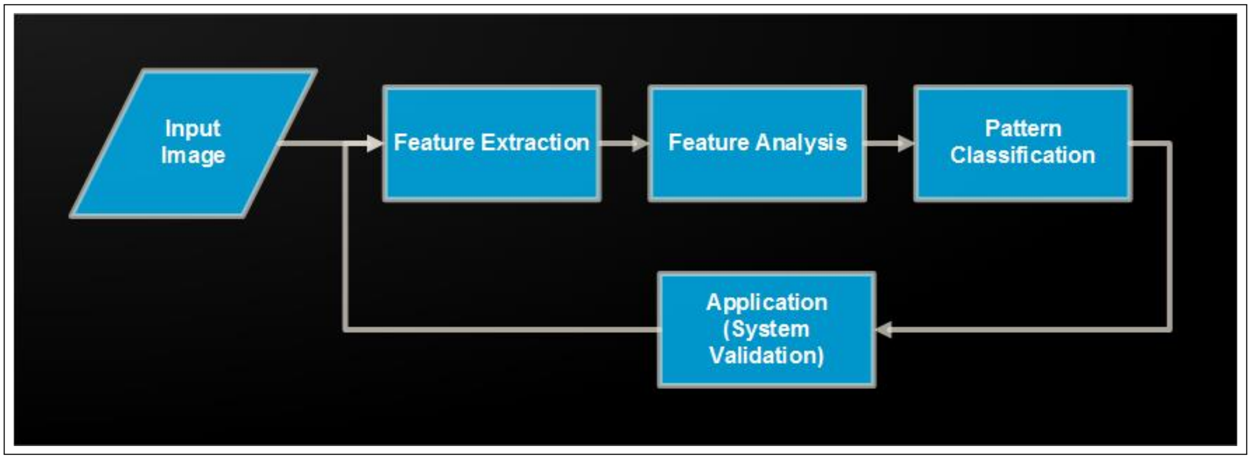


Figure 5.1: Illustrative diagram on Feature Analysis

where $m \geq n$ and $m,n \in Z^+$, $R^{r \times m}$ is any database or matrix containing the original feature set having r instances or observation, $R^{r \times n}$ is the reduced feature set containing r observations in the subset selection.

This is further illustrated in Figure 5.2. It is to be noted that Feature selection is inherently a multi-objective problem with two main objectives of minimizing both the number of features and classification error (O. Babatunde et al., 2014b).

Definition 5.1.2. Given a measure μ, and a dataset D with features $X_i, i(1)n$, from a distribution D over the labeled instance space, an optimal feature subset, $\mathrm{X_{opt}}$, is a subset of the features such that the accuracy of the induced Classifier $\mathrm{C} = \mu(\mathrm{D})$ is maximal (Kohavi & John, 1997).

Definition 5.1.3. A feature $\mathrm{X_i}$ is relevant iff $\exists$ some x_i and y for which $p(X_i = x_i) > 0$ such that $p(Y = y \quad | \quad X_i = x_i) \neq p(Y = y)$ (Kohavi & John, 1997).

Definition 5.1.4. A Feature is deemed redundant if one or more of the other features are highly correlated with it.

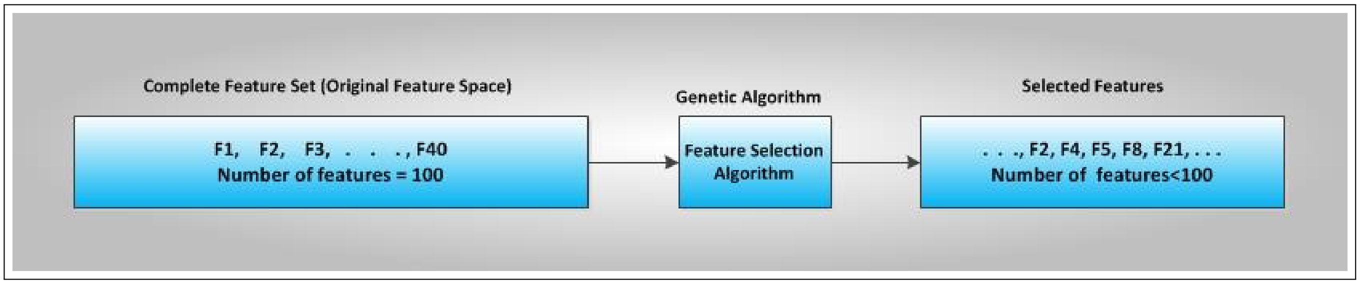

Complete Feature Set (Original Feature Space)
F1, F2, F3, . . . , F40
Number of features = 100
Genetic Algorithm
Feature Selection Algorithm
Selected Features
. . ., F2, F4, F5, F8, F21, . . .
Number of features<100

Figure 5.2: Illustrative diagram on Feature selection

In this study, both GA-based and PSO-based feature selection (a subspace or manifold projection techniques) are used to optimize the initial setting of the concerned classifiers, so as to obtain the 'optimal' subset of features. The performance of the concerned learning machines is further evaluated with the features selected from both GA (see section 5.3) and PSO (see section 6.3).

5.2 DataSet (Feature Space)

1. Features Generated from the Flavia Dataset:

   The source of images of leaves used in this study are images of leaves found in the Flavia dataset which is publicly available (Wu et al., 2007). The Flavia dataset is a constrained set of leaf images taken against a white background and without any stem present. The species in the dataset have a varying number of instances as shown (O. Babatunde et al., 2014a, 2014b). The dataset has 1907 images of 32 species of plants. The complete feature space for this work comprises of ZMs, FDs, Lengendre Moments, Hu 7 Moments, Texture, Geometrical properties and colour features which are extracted from the Flavia dataset as dicussed in chapter 4. The variables $F_i, i = 1(1)Num$, $Num = 100$ in Table 5.2 and $Num = 112$ in Table 5.1 represent the features extracted. Thus the feature space herein is a $\mathbb{R}^{r \times m}$ matrix, where r, (number of observations)$= 1907$ and m, (number of attributes or futures required)$= Num$.

Table 5.1: PhD Dataset 1 ↦ First features (100 non-colour features) derived from the Flavia Dataset

Observation	F1 F2 F3 F4 F5 F6 F7 F8 F9 ...F100
Image1	$X_{1,1}$ $X_{1,2}$ $X_{1,3}$ $X_{1,4}$ $X_{1,5}$... $X_{1,100}$
Image2	$X_{2,1}$ $X_{2,2}$ $X_{2,3}$ $X_{2,4}$ $X_{2,5}$... $X_{2,100}$
Image3	$X_{3,1}$ $X_{3,2}$ $X_{3,3}$ $X_{3,4}$ $X_{3,5}$... $X_{3,100}$
Image4	$X_{4,1}$ $X_{4,2}$ $X_{4,3}$ $X_{4,4}$ $X_{4,5}$... $X_{4,100}$
Image5	$X_{5,1}$ $X_{5,2}$ $X_{5,3}$ $X_{5,4}$ $X_{5,5}$... $X_{5,100}$
Image6	$X_{6,1}$ $X_{6,2}$ $X_{6,3}$ $X_{6,4}$ $X_{6,5}$... $X_{6,100}$
...	...
Image1907	$X_{1907,1}$ $X_{1907,2}$ $X_{1907,3}$... $X_{1907,100}$

Table 5.2: PhD Dataset 2 ↦ Second features (both colour and non-colour) derived from the Flavia Dataset

Observation	F1 F2 F3 F4 F5 F6 F7 F8 F9 ...F112
Image1	$X_{1,1}$ $X_{1,2}$ $X_{1,3}$ $X_{1,4}$ $X_{1,5}$... $X_{1,112}$
Image2	$X_{2,1}$ $X_{2,2}$ $X_{2,3}$ $X_{2,4}$ $X_{2,5}$... $X_{2,112}$
Image3	$X_{3,1}$ $X_{3,2}$ $X_{3,3}$ $X_{3,4}$ $X_{3,5}$... $X_{3,112}$
Image4	$X_{4,1}$ $X_{4,2}$ $X_{4,3}$ $X_{4,4}$ $X_{4,5}$... $X_{4,112}$
Image5	$X_{5,1}$ $X_{5,2}$ $X_{5,3}$ $X_{5,4}$ $X_{5,5}$... $X_{5,112}$
Image6	$X_{6,1}$ $X_{6,2}$ $X_{6,3}$ $X_{6,4}$ $X_{6,5}$... $X_{6,112}$
...	...
Image1907	$X_{1907,1}$ $X_{1907,2}$ $X_{1907,3}$... $X_{1907,112}$

2. Dataset from 100 plant species:

This dataset is reported in the paper by (Mallah, Cope, & Orwell, 2013). The dataset comprises a hundred species of leaves with each species having sixteen distinct specimens (observations), photographed as a colour image on a white background. The authors of this work provided only binary version of the original colored samples. This explains the reasons why only non-color features were extracted from this dataset.

Table 5.3: PhD Dataset 3 ↦ Non-colour features generated binary dataset

Observation	F1 F2 F3 F4 F5 F6 F7 F8 F9 ...F100
BinaryImage1	$X_{1,1}$ $X_{1,2}$ $X_{1,3}$ $X_{1,4}$ $X_{1,5}$... $X_{1,100}$
BinaryImage2	$X_{2,1}$ $X_{2,2}$ $X_{2,3}$ $X_{2,4}$ $X_{2,5}$... $X_{2,100}$
BinaryImage3	$X_{3,1}$ $X_{3,2}$ $X_{3,3}$ $X_{3,4}$ $X_{3,5}$... $X_{3,100}$
BinaryImage4	$X_{4,1}$ $X_{4,2}$ $X_{4,3}$ $X_{4,4}$ $X_{4,5}$... $X_{4,100}$
BinaryImage5	$X_{5,1}$ $X_{5,2}$ $X_{5,3}$ $X_{5,4}$ $X_{5,5}$... $X_{5,100}$
BinaryImage6	$X_{6,1}$ $X_{6,2}$ $X_{6,3}$ $X_{6,4}$ $X_{6,5}$... $X_{6,100}$
...	...
...	...
...	...
BinaryImage1600	$X_{1600,1}$ $X_{1600,2}$ $X_{1600,3}$... $X_{1600,100}$

3. Ionosphere dataset

   The alternative dataset used for testing this work is Ionosphere dataset from the University College London machine learning repository available at http://archive.ics.uci.edu/ml/datasets/Ionosphere. This dataset comprises of 351 observations and 34 attributes with binary class information (bad radar and good radar returns).

Table 5.4: Ionosphere dataset

Radar Observation	F1 F2 F3 F4 F5 F6 F7 F8 F9 ...F34
Observation 1	$X_{1,1}$ $X_{1,2}$ $X_{1,3}$ $X_{1,4}$ $X_{1,5}$... $X_{1,34}$
Observation 2	$X_{2,1}$ $X_{2,2}$ $X_{2,3}$ $X_{2,4}$ $X_{2,5}$... $X_{2,34}$
Observation 3	$X_{3,1}$ $X_{3,2}$ $X_{3,3}$ $X_{3,4}$ $X_{3,5}$... $X_{3,34}$
Observation 4	$X_{4,1}$ $X_{4,2}$ $X_{4,3}$ $X_{4,4}$ $X_{4,5}$... $X_{4,34}$
Observation 5	$X_{5,1}$ $X_{5,2}$ $X_{5,3}$ $X_{5,4}$ $X_{5,5}$... $X_{5,34}$
Observation 6	$X_{6,1}$ $X_{6,2}$ $X_{6,3}$ $X_{6,4}$ $X_{6,5}$... $X_{6,34}$
...	...
...	...
...	...
Observation 351	$X_{351,1}$ $X_{351,2}$ $X_{351,3}$... $X_{351,34}$

5.2.1 Problem Statement

From Table 5.1. The following optimization problem was solved in this Chapter.

Given that $n \in \mathbb{Z}^+ \ni 1 \leq n \leq 100$ and $\alpha \in \mathbb{R}^+ \ni 0 \leq \alpha \leq 100$,

where

(i) $n =$ number of features in the reduced feature set.

(ii) $\alpha =$ Classification error.

Find a subset of features $F_i \in$ Table 5.1 such that the objective α and n are minimized.

5.3 Genetic Algorithm (GA)

Genetic Algorithms (GA) can be defined as population-based and algorithmic search heuristic methods that mimic natural evolution process of man. (Holland, 1962; Melanie, 1999; Tian et al., 2012). GA iteratively employ the use of one population of chromosomes (solution candidates) to get a new population using a method of natural selection combined with genetic functionals such as crossover and mutation (in the similitude of Charles Darwin evolution principle of reproduction, genetic recombination, and the “survival of the fittest”). In comparative terminology to human genetics, chromosomes are the bit strings, gene is the feature, allele is the feature value, locus is the bit position, genotype is the encoded string, and phenotype is the decoded genotype (Sivanandam & Deepa, 2008). The fitnesses of the chromosomes are evaluated using a function commonly refered to as Objective function or fitness function. In other words, the fitness function (objective function) reports numerical values which are used in ranking the chromosomes in the population. The formulation of the fitness function depends on the problem being solved. Example of a fitness function is the parabola $y : x \mapsto ax^2 + bx + c$ which is used in minimizing or maximizing quadratic functions over admissible range of real or complex values. The triple {a,b,c} in this expression are constants. The GA in the Toolbox is a minimizer of objective function as opposed to many other GA which maximizes. However the maximization problem can be tuned to be a minimization problem via the user-defined fitness function. As an example, suppose the function $f(x) = x^2$ is to be minimized viz $min[f(x)]$, then the dual formulation of this problem is to maximize the negative of $f(x)$, written as $max[-f(x)]$. Thus maximizing the negative of a function is equivalent to minimizing its positive. In relation to this PhD study, maximizing classification accuracy is equivalent to minimizing error rate.

5.4 GA-Based Feature Selection

The five important issues in the GA are chromosome encoding, population initialization, fitness evaluation, selection (followed by genetic operators), and criteria to stop the GA (see Figure 5.3). The GA operates on binary search space as the chromosomes are bit strings. The GA manipulates the finite binary population in similitude of human natural evolution. First, an initial population is created randomly and evaluated using a fitness function. As regards binary chromosome used in this work, a gene value ’1’ indicates the particular feature indexed by the position of the ’1’ is selected. If it is ’0’, the feature is not selected for evaluation of the chromosome concerned. This is shown in Figure 5.4 which display 20 genes for 30 chromosomes for clarity purpose. Each row in Figure 5.4 is a chromosome containing genes valued as either 0 or 1. The chromosomes are then ranked and based on the rankings, the top n fittest kids (Elitism of size n) are selected to survive to the next generation. The fitness evaluation is done through Algorithm 4. After the elite individuals are moved to the next generation, the remainning individuals in the current population are used to produce the rest of the next generation through crossover and mutation. Crossover is basically, combination of two individuals to form a crossover kid (see Section 5.4.6). Mutation operator on the other hand, depicts a genetic pertubation of the genes in each chromosomes through flipping of bits depending on the mutation probability (see Section 5.4.7). Following the steps in Figure 5.3, the steps involved in using the GA for feature selection are explained in this section.

5.4.1 Generation of Initial Population

The initial population for this work is a matrix of dimension PopulationSize × ChromosomeLength containing only random binary digits. The PopulationSize is the number of chromosomes (individuals) in the population, while ChromosomeLength (GenomeLength) is the number of bits (genes) in each chromosome. It is a good idea to make the population size to be at least the value of the chromosome length so that the chromosomes in each population span the search space (MathWorks, 2013). The pseudocode for initial population is given in Algorithm (2).

Algorithm 2 Creation of Initial Population

```
procedure PopFunction()
    $pop \leftarrow$ Binary Matrix of size $PopulationSize \times GenomeLength$
    Return $pop$
end procedure
```

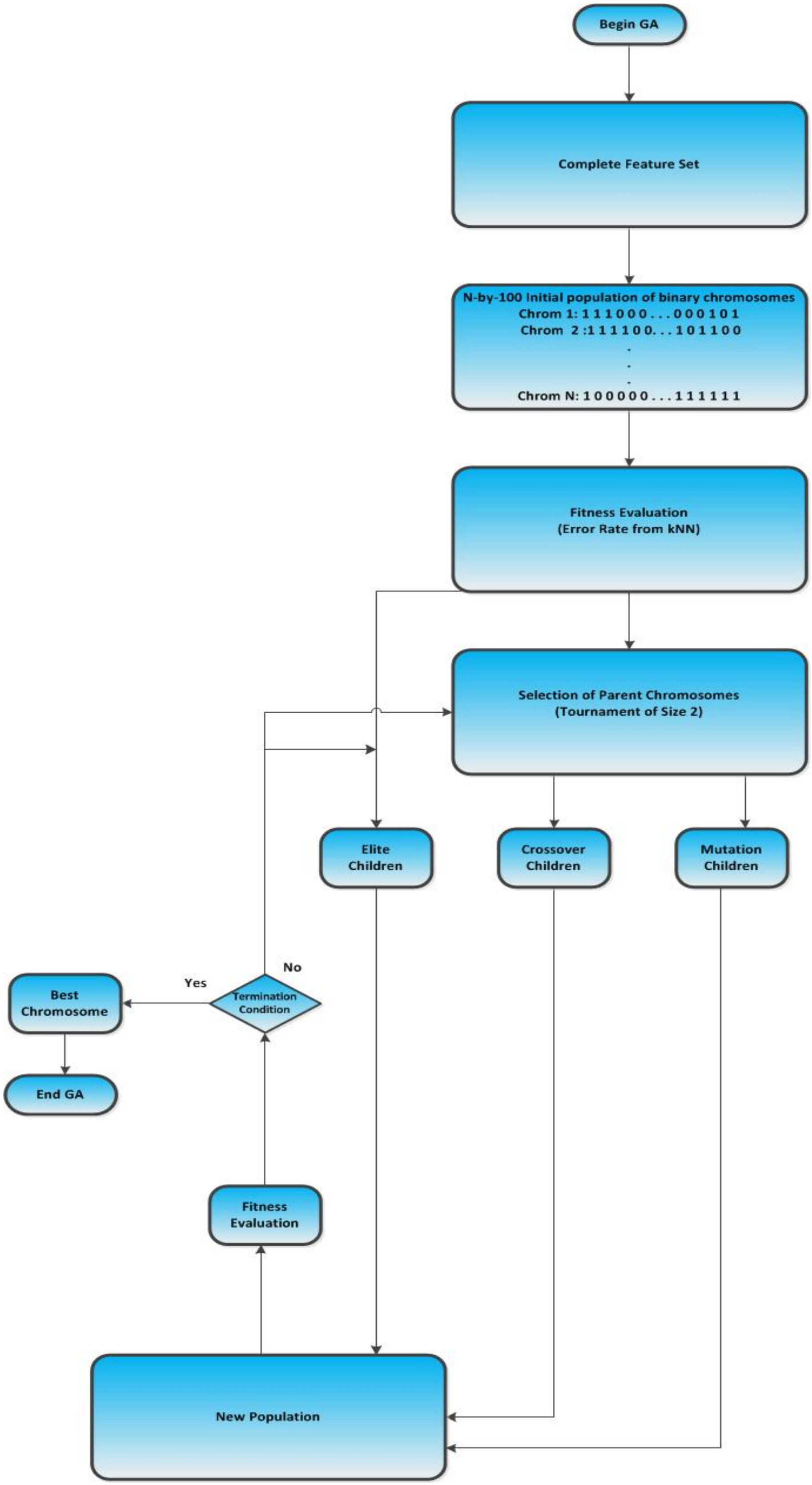
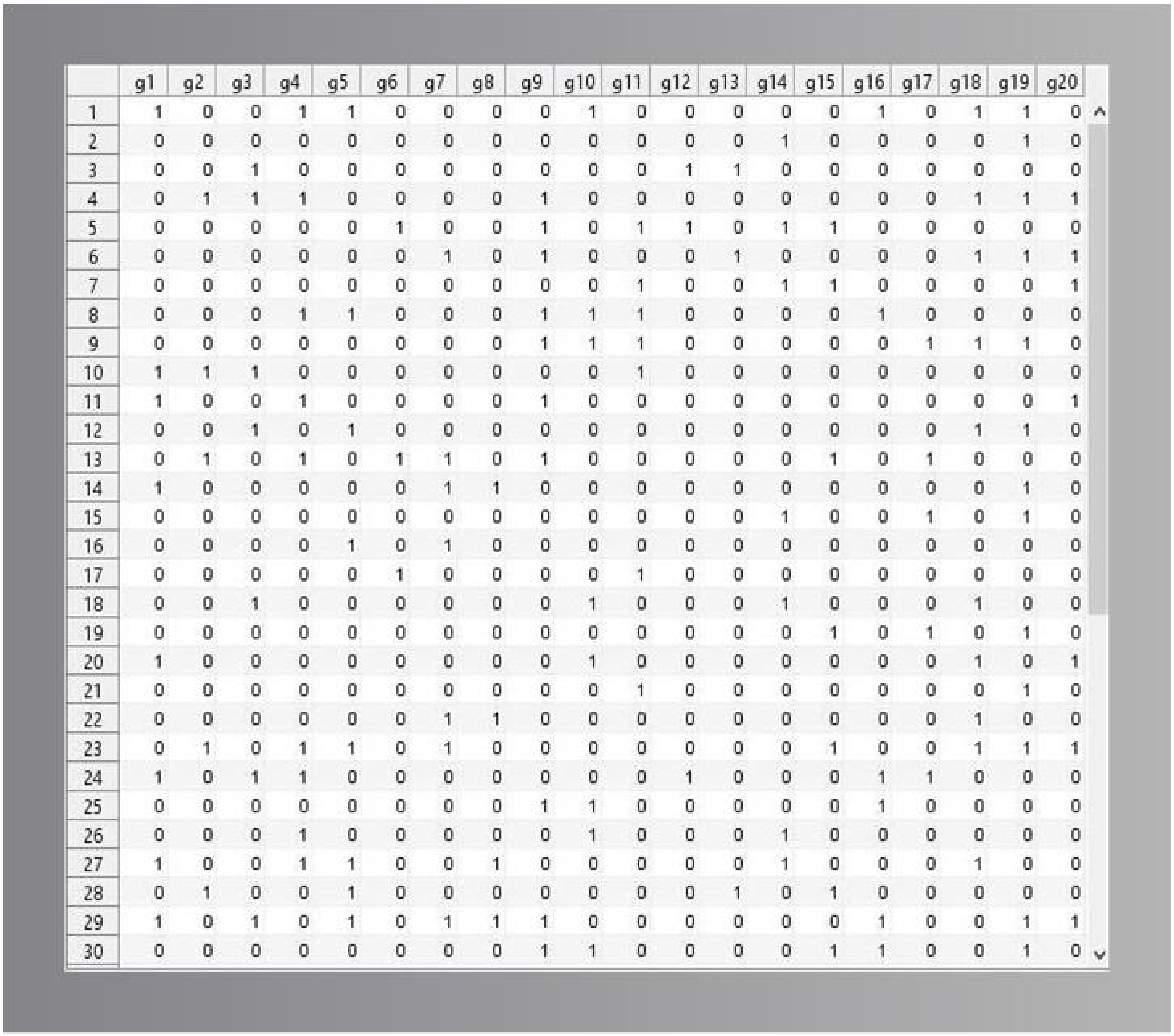


Figure 5.3: GA-Based Feature Selection



	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	g13	g14	g15	g16	g17	g18	g19	g20
1	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	1	0	1	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
3	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
4	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1
5	0	0	0	0	0	1	0	0	1	0	1	1	0	1	1	0	0	0	0	0
6	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	1	1	1
7	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	1
8	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0
10	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
11	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
12	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
13	0	1	0	1	0	1	1	0	1	0	0	0	0	0	1	0	1	0	0	0
14	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0
16	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
18	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0
20	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1
21	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
22	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
23	0	1	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	1	1	1
24	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0
25	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0
26	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
27	1	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0
28	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
29	1	0	1	0	1	0	1	1	1	0	0	0	0	0	0	1	0	0	1	1
30	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0

Figure 5.4: A sample showing initial binary population of chromosomes. The postional index of 1s in each row of the matrix represent the index of the features used in fitness evaluation. Those whose positional index are 0s are not used in the fitness evaluation

Table 5.5: Parameters Used in GA

GA Parameter	Value
Population size	100
Genomelength	100
Population type	bitstrings
Fitness Function	kNN and PNN-Based Classification Error)
Number of generations	300
Crossover	Arithmetic Crossover
Crossover Probability	0.8
Mutation	Uniform Mutation
Mutation Probability	0.1
Selection scheme	Tournament of size 2
EliteCount	2

5.4.2 Fitness Evaluation

For GA to select a subset of features, a fitness function (a driver for the GA) must be defined to evaluate the discriminative capability of each subset of features. The fitness of each chromosomes in the population are evaluated using kNN-based fitness function (see Algorithm 4). The PhD Dataset is in two-fold viz (PhD Dataset 1 (non-colour features) & PhD Dataset 2 (both non-colour and colour features) as shown in Table 5.1 and comprises of {100, 112} features respectively. The other Dataset (Ionosphere Dataset) contains 34 features. The kNN is useful for classification not requiring model building, and hence, it is called ”instance-based learning”. The kNN algorithm solves classification problem by looking for the shortest distance between the test data and training sets in the feature space (Cover & Hart, 1967). The distance is generally computed in Pythagorean sense (by finding the square root of the sum of differences). Suppose the training set, using the PhD and Ionosphere Dataset, is defined as:

$$\mathrm{x} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix}. \tag{5.4.1}$$

where M = {100, 112, 34} for the PhD Dataset 1, PhD Dataset 2 and Ionosphere Dataset respectively. The M is the number of observations in these dataset. Each x_i (i = 1(1)100, 112) is a vector containing {100, 112} features as shown in Table 5.1. The kNN algorithm computes Euclidean distance between test data x_{test} and the training sets and then find the nearest point (shortest distance) from the training set to the test set as:

$$D(x_{test}, x_i) = \sqrt{\sum_{m=1}^{M} (x_{test} - x_{im})^2} \tag{5.4.2}$$

As usual , the kNN in the fitness function considers only the k nearest neighbors (local information) denoted by $x_1, \ldots, x_k$ as the member(s) of the set (a normed linear space)

$$kNNSpace = \{x_j | d(x, x_i) \leq d(x, x_i)\} \tag{5.4.3}$$

The kNN rules involves classifying a test sample, say, x by assigning it the most frequently represented among the k nearest samples. The diagram in Figure 5.5 taken from (MathWorks, 2013) illustrates 3 Nearest Neighbors as they are the three shortest distances reported. A similar figure generated from this study, showing first 18 neighors of a test sample from the Flavia dataset is shown in Figure 5.6.

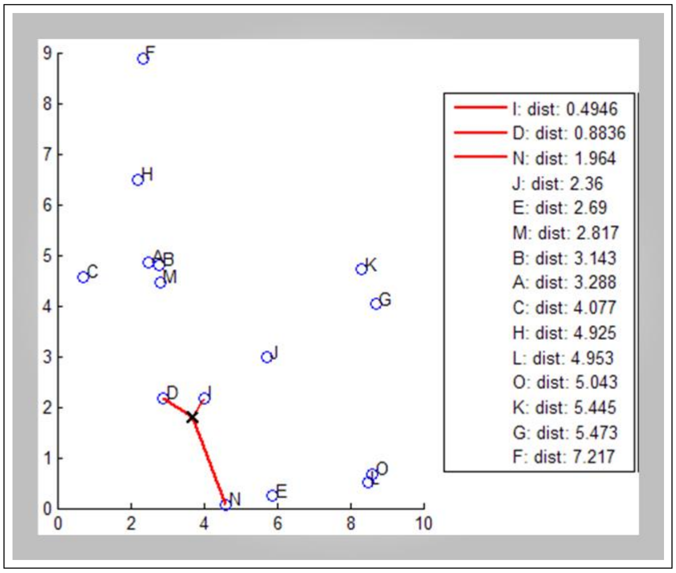


Figure 5.5: A diagram showing k = 3 nearest neighbors (Mathworks, 2013)

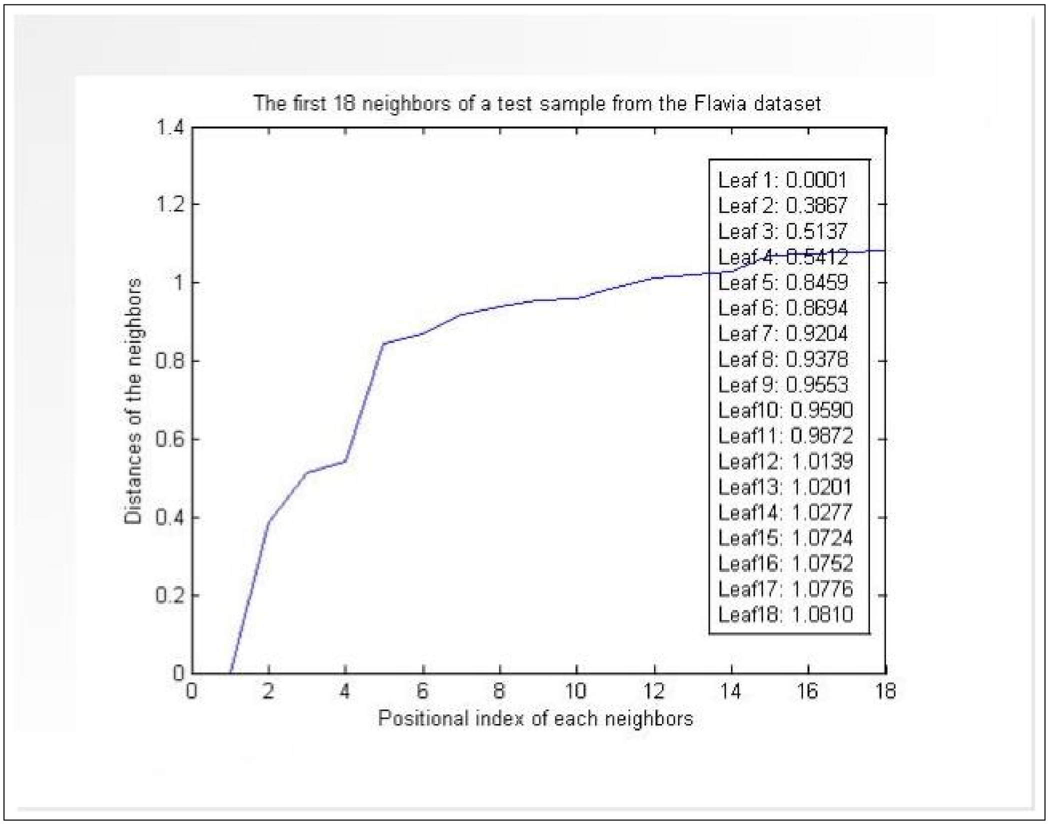


Figure 5.6: A diagram showing k = 18 nearest neighbors

Definition 5.4.1. A normed linear space, say, kNNSpace $(E, \| \,.\, \|)$ is a metric space that is endowed with the following properties

1. $\| x \| \geq 0$ for all $x \in E$ and $\| x \| = 0$ if and only if $x = 0$

2. $\| \alpha x \| = |\alpha| \, \| x \|$ for all $x \in E$ and $\alpha \in \mathbb{R}$.

3. $\| x + y \| \leq \| x \| + \| y \|$ for all $x \in E$ (triangular inequality)

Since $k = 3$ in this research the kNN count each category m in the class information (accumulated as $count(x_m)$) using 3 Nearest Neighbors and then report classification results based on the expression

$$argmax(count(x_m)) \tag{5.4.4}$$

subject to

$$\sum_{i=1}^{M} count(x_m) = class \tag{5.4.5}$$

where $class$ = {1(1)32, 1(1)2} for the PhD Dataset and Ionosphere Dataset respectively. Algorithm 3 summarizes the operation of the KNN used. The classification accuracy of kNN algorithm is sensitive to the value of k.

In each chromosome a gene value '1' indicates the particular feature indexed by the position of the '1' is selected. If it is '0', the feature is not selected for evaluation of the chromosome concerned (see Figure 5.4). The genome (or chromosome) are the encoded bit strings represeting the features. As the GA iterates, the individuals (combinatorial set of features) in the current population are evaluated and their fitness are ranked based on the kNN-based classification error. Individuals with lower fitness have better chance of surviving into the next generation or mating pool. This ensures the GA reduce the error rate and picks the individual with the least (best) fitness value.

Algorithm 3 Algorithm for k Nearest Neighbor

```
procedure ComputekNN()
    Input $TRAININGSET = \{x_i, c_i\}$,$x = featureset, i = 1(1)M, M =$ number of observations in the training set, $c =$ class information, $j = 1(2)N_c$, $N_c$ = number of classess available.
    Assign $p_i \leftarrow \{x_i, c_i\}, i = 1(1)M, c_i \in N_c$.
    Compute $D(x_{test}, x_i) = \sqrt{\sum_{m=1}^{M} (x_{test} - x_{im})^2}$
    sort $p_i$ based on D.
    Select the first k points from the sorted list
    Assign ClassLabel $\leftarrow p^*$ if $c^* = argmax(count(x_i))$
end procedure
```

$$FitFunc1 = \frac{\alpha}{N_T - N_f} \tag{5.4.6}$$

where

α = kNN-Based classification error.

N_T = Cardinality of the original feature set

N_f = Cardinality of the selected features.

Algorithm 4 Fitness Function Evaluation

```
procedure FitFunction1()
    FeatIndex ← Indices of ones from BinaryChromosome
    NewDataSet ← DataSet indexed by FeatIndex
    NumFeat ← Number of elements in FeatIndex
    3 ← NumNeighborskNN
    $kNN_{Error}$ ← $Classifier_{KNN}$(DataSet,ClassInformation,NumNeighborskNN)
    Return $kNN_{Error}$
end procedure
```

5.4.3 Generation of Children for New Population

After fitness evaluation, new population is created using Elitism and Genetic Operators (Crossover and Mutation). In this GA, three types of children are created to form the new population. They are:

(a) Elite children: These children are given pushed automatically into the next generation (being those with the best fitness values). Elitism in the GA Toolbox is specified by the identifier ”EliteCount” with default value of 2. It is obviously bounded by the population size. This implies ”EliteCount” $\leq PopulationSize$. With size 2, GA picks the top two best chromosomes and push them automatically to the next generation.

(b) Crossover Children: This is explained in section (5.4.6).

(c) Mutation Children: This is explained in section (5.4.7).

5.4.4 Proportion of Elite, Crossover, and Mutation Children in the New Population

Table 5.5 shows the configuration of the GA in this work. From Table 5.5, the length of each chromosome for the PhD Dataset is 100 since we have a total number of 100 non-colour features extracted from the Flavia dataset. The maximum number of generation was set to 300 to avoid the GA been trapped in local optimal.

To create new population, the GA performs Elitism, Crossover and Mutation in sequential order.

(1) Elite Children: The number of elitism as shown in Table 5.5 is 2. Therefore, the top two kids with the lowest fitness values are automatically pushed in the next generation.
Thus,

Number(Elite kids) = C_1 = 2. This means there are 98 (i.e 100-C1) individuals in the population apart from elite kids. From the remaining 98 chromosomes, crossover and mutation kids are then produced.

(2) Crossover Children: The proportion (fraction) of the next generation, apart from the left over kids, that are produced by crossover is called Crossover fraction. Crossover fraction used in this work is 0.8. If this fraction is set to one, then there is no mutation kids in the GA, otherwise, there will be mutation kids. With the fraction taken as 0.8, then the number of crossover children will be $C_2 = \text{round}(98 * 0.8) = 78$

(3) Mutation Children: Finally, number of mutation children is:

C_3 = 100- C_1 - C_2 = 100-78-2 = 20

This implies $C1 + C2 + C3 = 100$

When the genealogical plots of each individuals are made, mutation children are indicated by red lines, crossover children by blue lines, and elite children by black lines.

5.4.5 Selection Mechanism Used: Tournament

The aim of selection mechanism in GA is to make sure the population (solution candidates) is being constantly improved over all fitness values. The selection mechanism helps the GA in discarding bad designs and keeping only the best individuals. There are many selection mechanism in the GA Toolbox, the default of this being stochastic uniform (with default size 4) but we have decided to use Tournament Selection of size 2 due to its simplicity, speed and efficiency(Eiben & Smith, 2010). Also, tournament selection enforces higher selection pressures on the GA(resulting in higher rate of convergence) and makes sure the worst individual does not get into the next generation (Sivanandam & Deepa, 2008; O. Babatunde, Armstrong, Leng, & Diepeveen, 2014c). In the GA implemented for this study, two functions are needed to perform tournement selection. The first function generates the players (parents) needed in the actual tournament function, while the second function which outputs the winner of the tournament. A counter is set which runs from 1 to the number of chromosomes in the playerlist. The fitnesses of the selected chromosomes are ranked and the best of this becomes the winner. In tournament selection of size 2, two chromosomes are selected from the population after the Elite kids are taken out and the best of the two chromosomes,(using fitness ranking), is selected. Tournament selection is performed iteratively until the new population is filled up.

5.4.6 Crossover function

The crossover operator in the GA genetically combines two individuals (parents) to form children for the next generation. Two parents chromosomes are needed to carry out crossover operation. The two chromosomes are taken from tournament selection. The GA uses crossover fraction, say, XoverFrac to specify the number of kids produced by the crossover functional after Elite kids are removed from the current population being evaluated.. The variable XoverFrac, as discussed in the preceding section, is bounded by the inequality $0 \leq XoverFrac \leq 1$. The value used for XoverFrac in this work is 0.8 and the crossover function chosen is arithmetic type. In this case , XOR operation is performed on the two parent chromosomes since they are binary (Siddique & Adeli, 2013; Marek, 1998; MathWorks, 2013). This is illustrated in Equation 5.4.7.

$$CrossOverKids(ii) = p1 \bigoplus p2 \tag{5.4.7}$$

where ii is an index that runs from 1 to the number of kids needed for crossover;
$\bigoplus$ is an XOR operator for binary operands;
$p1$ = first parent needed by the crossover function;
$p2$ = second parent needed by the crossover function;
The XOR operator $\bigoplus$ works as follows:

$1 \bigoplus 1 = 0$
$1 \bigoplus 0 = 1$
$0 \bigoplus 1 = 1$
$0 \bigoplus 0 = 0$

So for two binary operands (parent chromosomes) viz;
p1 = 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1
p2 = 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 0 1 1
CrossOverKid = p1 $\bigoplus$ p2
CrossOverKid = 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0

5.4.7 Mutation function

Mutation is a genetic pertubation of individuals in the population. Mutation ensures genetic diversity and searching of broader solution space. We used uniform mutation as our choice. For uniform mutation, the GA generates GenomeLength set of random numbers (RDs) from uniform distribution. The value of each random number is associated with the position of each gene (bit) in the chromosome. The chromosome is scanned from left to right and for each associated bit, the value of each RD is compared with the mutation probability (denoted as p_m) and if the RD at position i is less than p_m, then gene (bit) at position i is flipped. Otherwise, the gene is left unflipped. This is repeated from the Least Significant Bit (LSB) to the Most Significant Bit (MSB) of each chromosome in the mutation children.

As an example, given a parent chromosome p = 1 1 0 1 1 1 0 1 1 0 0 0 1 0 0 1 0 1 1 0;

The number of bits in this p is 20.

Therefore, a set of 20 uniform random numbers are generated, say,

RD20 = [0.5159 0.4161 0.5830 0.5138 0.2839 0.3934 0.2659 0.3776 0.9710 0.2595 0.1807 0.2244 0.5224 0.3166 0.4452 0.4138 0.3362 0.4145 0.5863 0.6220]

If the mutation probability is $p_m = 0.3$, then p_m is compared with each vector entry in RD20 and the following binary vector p_{mpoint} is obtained.

p_{mpoint} = 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0.

The positional index of 1s in p_{mpoint} are 5, 7, 10, 11, 12.

Finally, all the genes in p at locus 5, 7, 10, 11, 12 are flipped, which will then produce a mutation kid as:

MutKid = 1 1 0 1 0 1 1 1 1 1 0 1 1 0 0 1 0 1 1 0.

5.4.8 New Population (Member of next generation)

The GA iterates until the new population is filled up. The new population is filled by adding individuals from Elite kids, crossover kids, and mutation kids. This is illustrated by Equation 5.4.8.

$$\text{NewGeneration} = \boldsymbol{Number}(\text{Elit kids}) + \boldsymbol{Number}(\text{Crossover kids}) + \boldsymbol{Number}(\text{Mutation kids}). \quad (5.4.8)$$

This is then evaluated again (see Equation 5.4.9) and the selection-reproduction steps are repeated untill the stopping condition is met.

$$\text{NewGenerationScore} = \boldsymbol{Fitness}(\text{NewGeneration}) \qquad (5.4.9)$$

5.4.9 Repeat Until GA Termination Conditions Occur

Once the GA reaches optimum solution, it stops. The code condition at which the GA stops is called stopping conditions. The two stopping conditions applicable to this work are:

(a) Maximum Number of Generations ('Generations' $\in \boldsymbol{Z}^{+}$).
(b) Stall Generation Limit ('StallGenLimit' $\in \boldsymbol{Z}^{+}$).

The GA can terminate prematurely if the 'Generations' is not properly set. The value of 'Generations' for this work was set to be 300 while the value 100 was used for StallGenLimit. The GA terminates if the average change in the fitness values among the chromosomes over StallGenLimit generations is less than or equal to Tolfun which is valued as 0.000001. Specifically, the GA examines the difference in values of fitnessess of all generations and if the average of these differences for 100 generations is less than or equal to 0.000001, the GA terminates. This implies genetic homogeneity (similarity in fitness values) among the chromsomes of the generation containing the best chromosome and consequently, the convergence of the GA.

5.5 Simulation and Experimental Results

Based on the GA configuration in Table 5.5, the following results were obtained. The carefully chosen fitness function enabled the GA to appreciably minimizes number of features needed and classification error from kNN. As a proof (from MATLAB documentation), the best fitness and mean fitness should be close in value as the GA iterates. The stall generation is number of generations produced by the GA since the last upgrade of the fitness value. The GA terminates at generation 101. This is also evidenced by the GA simulation diagrams in Figures (5.7, 5.8, & 5.9) where the value of the fitness function remains constant from generation 58 to 101 and generation 38 to 101 for the PhD Dataset and Ionosphere Dataset respectively. The features selected by the GA from both datasets are:

1. Selected Feature 1 (PhD Dataset (Non-colour features)) = [1, 6, 12, 15, 18, 56, 64, 71, 72, 75, 78] i.e (11% of the original dataset). Even using mutual information as the fitness function, the features obtained were : [1, 6, 10, 15, 26, 44, 53, 73, 74, 75]. Using the GA on the extracted color features, the binary string ColourString = [1 1 1 0 0 0 0 0 0 1 1 1] was obtained. This means the colour features associated with with the positional index [1, 2, 3, 10, 11, 12] were obtained.

2. Selected Feature 2 (PhD Dataset (Non-colour + colour features)) = [3, 6, 8, 21, 56, 71, 75, 78, 101, 102, 103, 110, 111, 112] i.e 12.50% of the original dataset which 112 in number

3. Selected Feature 3 (Ionosphere Dataset) = 2, 3, 5, 6, 7, 8, 27, 34 i.e (24% of the original dataset).

The best and mean fitness value, using the kNN classification error for PhD DataSet were 0.1746 and 0.181 while that of Ionosphere Dataset were 0.06268 and 0.07151. These results are validated in Section 5.7.

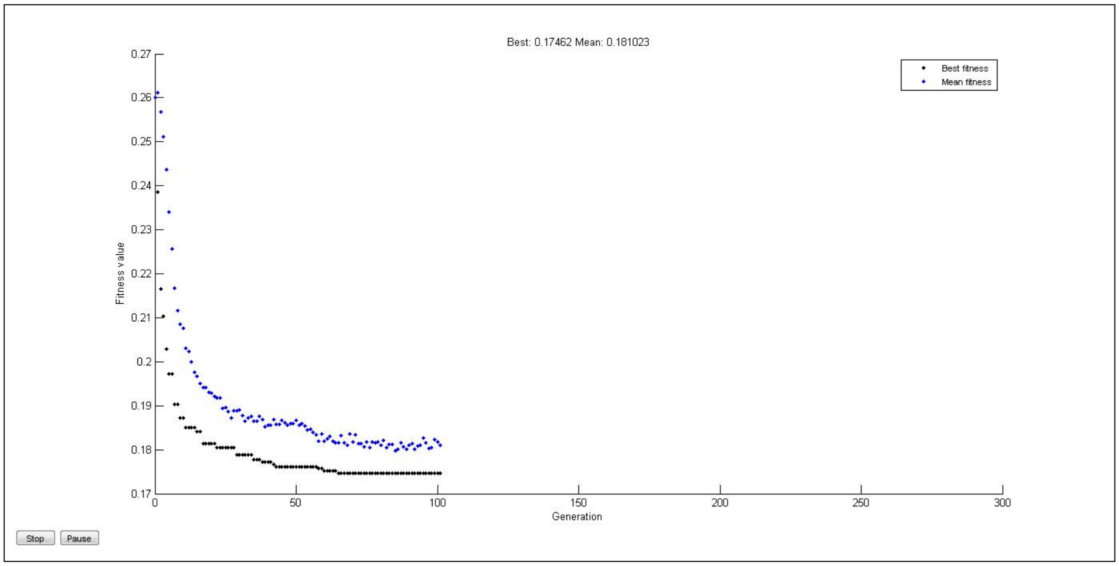


Figure 5.7: GA Simulation Diagram on PhD DataSet (non-colour features)

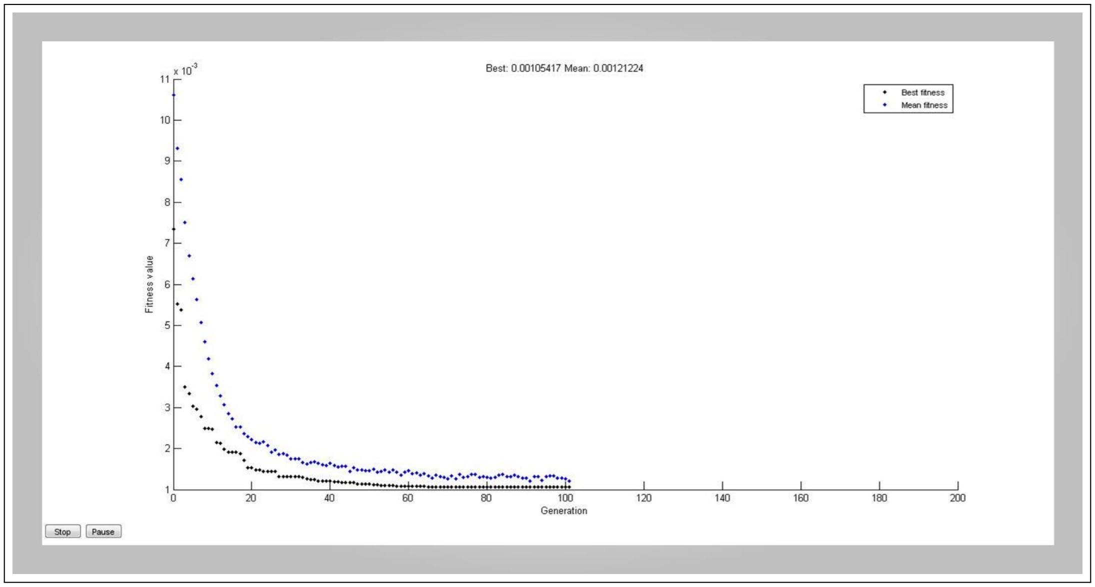


Figure 5.8: GA Simulation Diagram on PhD DataSet (non-colour + colour features)

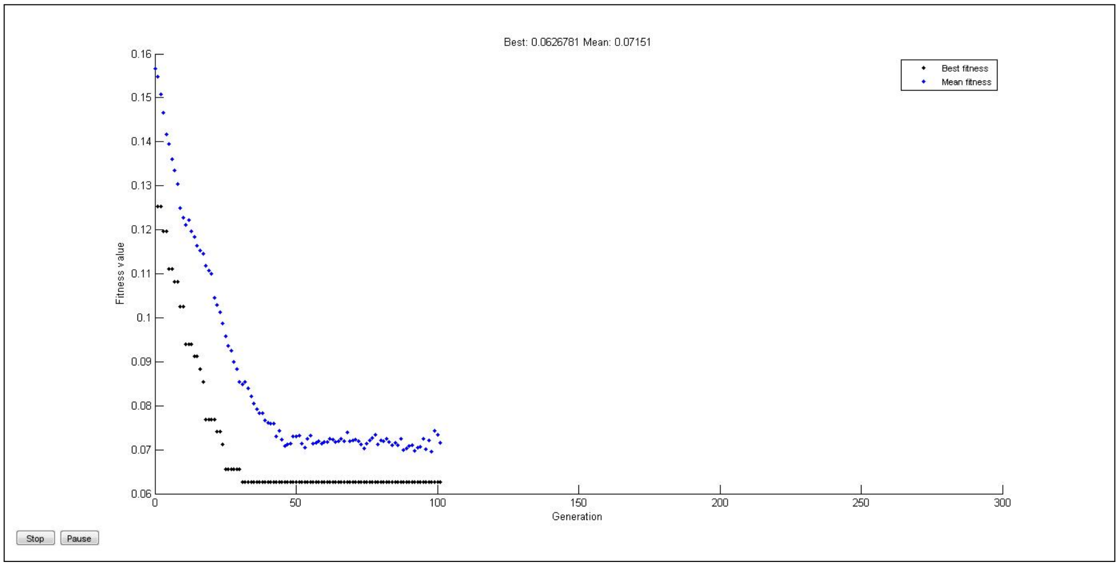


Figure 5.9: GA Simulation Diagram on Ionosphere DataSet

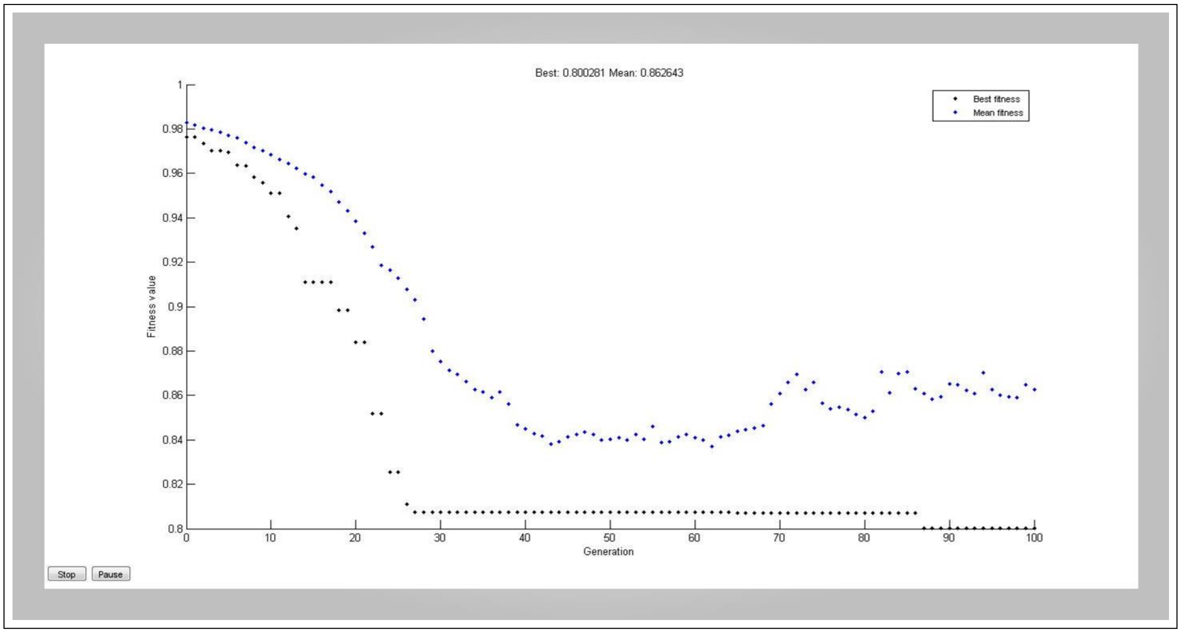


Figure 5.10: GA Simulation Diagram on Dataset from 100 plant species

5.6 Multi-Objective Genetic Algorithm (MOGA)

MOGA was also used secondarily to select some features and those features selected are those indexed by the vector [15 17 74 75 78 101 102 103]. The rational for using MOGA is to further examine which of the feaures can survive by strictly and simultaneously reducing error rates and number of needed features. The MOGA may be described as follows:

$$min\ \ f(\theta) = min\ \ [f_1(x), f_2(x), f_3(x), ..., f_n(x)] \tag{5.6.1}$$

subject to :

$$h_j(x) \quad \leq \quad 0 \quad , \quad (1 \leq j \leq K) \tag{5.6.2}$$

$$hh_j(x) \quad = \quad 0 \quad , \quad (1 \leq j \leq K) \tag{5.6.3}$$

$$x_{LOWER} \quad \leq \quad x_i \quad \leq \quad x_{UPPER} \tag{5.6.4}$$

where $f_i(x), i \in \Omega, i = 1(2)n$, are the objectives to be optimized , x is a solution and $h_j(x)$, x_{LOWER} & x_{UPPER} are the constraints and solution limits imposed on the problem. To solve the MOGA, the pareto optimal set x_p must be found. The pareto set are the solutions where none of the candidates dominates any of the others.

A solution x^1 dominates another solution x^2, denoted by $x_1 \prec x_2$, $\Longleftrightarrow$

$$\forall i \quad \in \Omega, \quad f_i(\theta^1) \quad \leq \quad f_i(\theta^2) \quad \wedge \quad \exists \quad \omega \in \Omega : \quad f_\omega(x^1) \quad \leq \quad f_\omega(x^2) \tag{5.6.5}$$

The pareto optimal set x_p is therefore given by :

$$\Theta_p = \quad \{x \in \mathrm{R} \quad | \quad \nexists \quad \tilde{x} \quad \in \mathrm{R} : \quad \tilde{x} \quad \prec \quad x\} \tag{5.6.6}$$

The set Θ_p is unique and normally includes infinite solutions. The R is the solution space. The MOGA-based selected features from this work are Zernike Moments (ZM) , Geometric features (GF) and mean of the RGB components of the images. This is an indication that both ZM, GF and RGB components are good candidates for building image classification systems. The diagrams in Figure 5.11 shows the parento font, chromosome fitness, rank histogram, and distance of chromosomes. Each points in the parento font are valid solution and equally optimal for the

MOGA and we have decided to pick those common to both GA and MOGA.

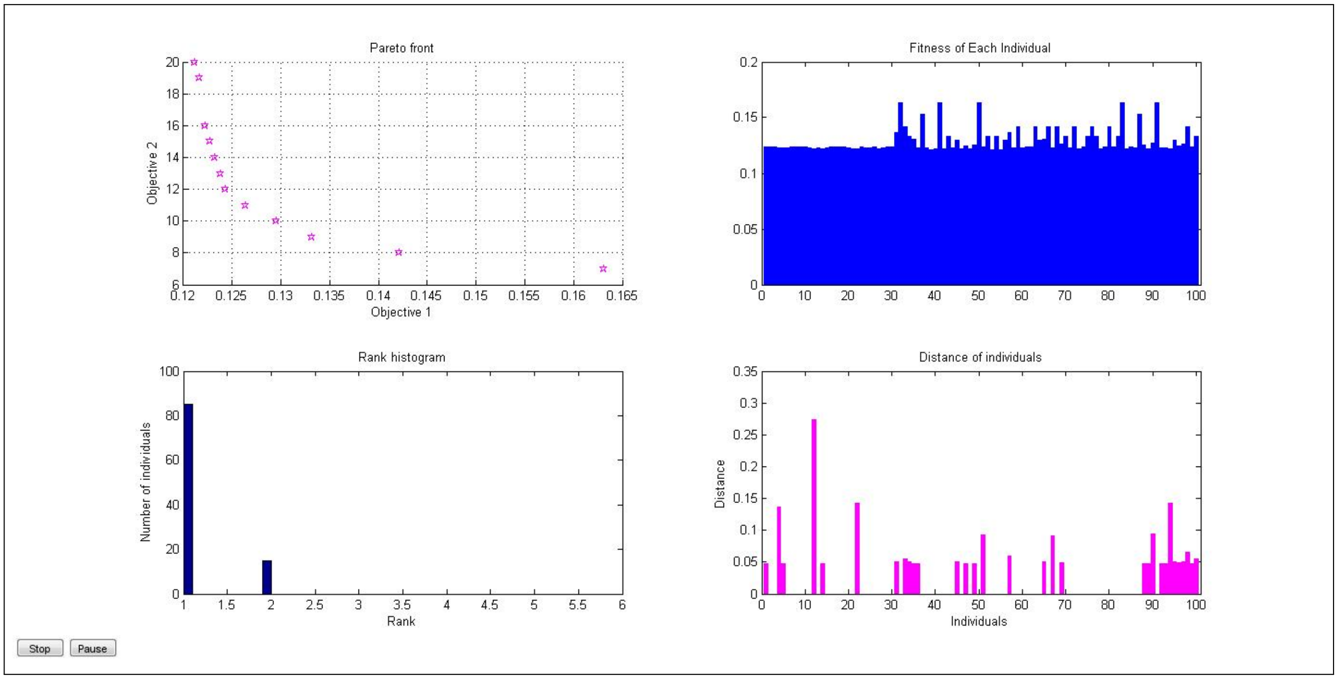


Figure 5.11: Multi-Objective GA Simulation Diagram on dataSet

5.7 Validation of Experimental Results

To validate the GA-FS in this work, the results were compared with a number of WEKA-Based feature selectors and the selected features were tested using a number of WEKA classifers such as Multi-Layer Perceptron (MLP), Random Forest (RF), J48, Naive Bayes (NB), and Classification using regression (RC). The two WEKA feature evaluators used are WEKA Correlation Feature Selection Subset Evaluator (WEKA CFS-SE) and WEKA ranker (Information Gain). The GA was evaluated using fitness function shown in Equations (5.4.6). The simulation diagram (Figures 5.7 & 5.9) based on the chosen fitness function shows convergence of the GA. The feature indexed by 6 in the PhD Dataset cuts across all the selectors. This proves a point that this feature will be useful for our classification system. This feature is Zernike Moment of order 6 and iteration 0 (i.e ZMI(6,0)).

When the PhD Dataset (non-color dataset) was fed into WEKA CFS which is a wrapper-based feature selector, 20 features were reported and this include the feature indexed by 6 also. Using the non-colour dataset, the GA interestingly selected 11 features ([1, 6, 12, 15, 18, 56, 64, 71, 72, 75, 78]) that were also selected by the WEKA ranker and CFS. The features common to both methods are Zernike moments, Hu moments, Texture properties, and Geometric properties. The GA approach has high level of controllability as the parameters in the GA configuration table can still be fine-tuned to obtain better results. Geometric properties of the Flavia dataset index by the vectors [70, 71, 72, 73, 74, 75, 76, 77, 78, 79] were also selected at rate more than any other features. The third features prefentially selected by all the selectors is Hu 7 moments (indexed by vectors [41, 42, 43, 44, 45, 46, 47]).

As can been seen from Table 5.8 and Table 5.9, the GA approach outperformed WEKA approach in some instances while WEKA also outperformed GA in some instances. In most cases, the difference in the classification accuracy reported by the two approaches are very small. The features selected by both method on the second dataset (ionosphere data) are shown in Table 5.7. In overall, both the GA method and WEKA-CFS which are wrapper-based feature selectors produced better classification accuracy than WEKA ranker (IG) which is filter-based. Since another chaper (chapter 6) discusses detailed comparative analysis of GA, PSO, and PCA, the features used in this section was only based on the non-color features as a matter of choice. The main advatange of the GA method lies in the area of controllability as the GA can be fine-tuned to produce better results all the time by changing the fitness functions.

Table 5.6: Comparison GA-FS with WEKA Feature Selectors Using first dataset (non-colour features)

S/N	Feature Selector	Selected Features
1	GA	1, 6, 12, 15, 18, 56, 64, 71, 72, 75, 78
3	WEKA (Information Gain Ranking Filter)	70, 77, 74, 73, 8, 50, 51, 66, 2, 9, 71, 48, 21, 61, 60, 62, 31, 79, 72, 6
4	WEKA (CFS Subset Evaluator)	1, 2, 6, 7, 12, 15, 16, 41, 43, 45, 51, 65, 66, 70, 71, 73, 74, 75 ,76 , 77

Table 5.7: Comparison GA-FS with WEKA Feature Selectors Using ionosphere dataset

S/N	Feature Selector	Selected Features
1	GA	2, 3, 5, 6, 7, 8, 27, 34
2	WEKA (Information Gain Ranking Filter)	5, 6, 33, 29, 3, 21, 34, 8, 13, 7, 31, 22
3	WEKA (CFS Subset Evaluator)	1, 3, 4, 5, 6, 7, 8, 14, 18, 21, 27, 28, 29, 34

Table 5.8: Classification Accuracy Using GA and WEKA-Based Features on first dataset (PhD Dataset)

S/N	Selector + WEKA Classifier	Accuracy(%)	RMSE
1	GA+MLP	72.88	0.1105
2	GA+RF	72.37	0.1037
3	GA+J48	67.97	0.1300
4	GA+NB	56.72	0.1222
5	GA+RC	69.70	0.1012
6	WEKA(IG)+MLP	73.52	0.1135
7	WEKA(IG)+RF	74.00	0.1065
8	WEKA(IG)+J48	70.84	0.1267
9	WEKA(IG)+NB	61.25	0.1483
10	WEKA(IG)+RC	74.62	0.1092
11	WEKA(CFS SE)+MLP	76.25	0.1084
12	WEKA(CFS SE)+RF	81.48	0.0961
13	WEKA(CFS SE)+J48	73.36	0.1223
14	WEKA(CFS SE)+NB	71.26	0.1268
15	WEKA(CFS SE)+RC	78.81	0.1017

Table 5.9: Classification Accuracy Using GA and WEKA-Based Features on second dataset (Ionosphere)

S/N	Selector + WEKA Classifier	Accuracy(%)	RMSE
1	GA+MLP	93.02	0.2339
2	GA+RF	94.35	0.1045
3	GA+J48	91.70	0.2462
4	GA+NB	90.55	0.2989
5	GA+RC	91.89	0.1988
6	WEKA(IG)+MLP	93.73	0.2388
7	WEKA(IG)+RF	92.59	0.2414
8	WEKA(IG)+J48	91.74	0.278
9	WEKA(IG)+NB	86.32	0.3387
10	WEKA(IG)+RC	89.74	0.2801
11	WEKA(CFS SE)+MLP	92.31	0.2499
12	WEKA(CFS SE)+RF	92.88	0.2427
13	WEKA(CFS SE)+J48	90.60	0.2982
14	WEKA(CFS SE)+NB	92.02	0.2682
15	WEKA(CFS SE)+RC	90.88	0.2666

5.8 Conclusion

In this chapter, a GA-based feature selection technique was described. The technique developed herein involved the use of a novel fitness function to select combinatorial set of features from original feature set. For benchmarking, features selected by both WEKA Feature Selectors and the GA were fed into a number of WEKA classifiers. The GA-based features outperformed WEKA-based features in more instances. In most cases, the difference in the classification accuracy reported by the two approaches are very small. The features selected by both method on the first and second dataset respectively, are shown in Tables 5.6 & 5.7. Overall, both the GA method and WEKA-CFS which are wrapper-based feature selectors produced better classification accuracy than WEKA ranker (IG) which is filter-based.

The main advantange of the method herein lies in the area of controllability as the GA can be fine-tuned to produce better results all the time by changing the fitness functions. Of all the features selected, ZM had the highest frequency followed by geometric features. The approaches in this work is much more promising than those from previous works such as (Wu et al., 2007) & (Urilch, 2010) as more features are extracted from the images used. With the application of GA for dimensionality reduction, more disciminating features were obtained. In the next chapter, GA-based feature selected is compared with two other common dimensionality reduction techniques which are Particle Swam Optmization (PSO) and Principal Component Analysis (PCA). This is to further validate the capability of GA in feature selection.

Chapter 6

Comparative Analysis of GA, PSO, and PCA

6.1 Introduction

This chapter details the exploration and application of Genetic Algorithm (GA) and Particle Swam Optimization (PSO) for the wrapper-based feature selection. A diagram showing a generic wrapper-based approach for feature selection is shown in Figure 6.1. Particularly a comparative study is carried out, examining the performances of both GA and PSO with respect to classification accuracy of some classifiers. 112 features were extracted features from set of images found in the Flavia dataset (a publicly available dataset). The extracted features, as discussed earlier in Chapter 4, are Zernike Moments (ZM), Fourier Descriptors (FD), Legendre Moments (LM), Hu's Moments (Hu7M), Texture Properties (TP), Geometrical Properties (GP), and Colour features (CF).

The main contribution of this chapter includes the comparison of two major optimization techniques, i.e., GA and PSO, and also their comparison with principal component analysis (PSO). The novel fitness function used in the evolutionary algorithms enabled both the GA and PSO to obtain a combinatorial set of feature giving rise to optimal accuracy. The effectiveness of these manifold projection techniques were tested on Probabilistic Neural Networks (PNN), k Nearest Neighbour (kNN) and Multilayer Perceptron (MLP). The experimental analysis demonstrates the classification accuracy with GA-based approach outperforming that with

PSO-based method. Since 14 features were selected by both GA and PSO respectively, the PCA herein was thus made to select the first 14 principal component. This chapter has been published in the two journal articles (O. Babatunde, Armstrong, Leng, & Diepeveen, 2015a, 2015b)

6.2 Genetic Algorithm (GA)

The GA was comprehensively dealt with in Chapter 5 since it's the major focus in this study regarding feature selection. Figure 5.3 illutrates the operation of the GA. The features generated by GA are those indexed by the following positional vectors:
$FeatVect1 = [3, 6, 8, 21, 56, 71, 75, 78, 101, 102, 103, 110, 111, 112]$

These features satisfy the definitions 5.1.2 & 5.1.3. Those not selected by the GA are those that satisfy the definition 5.1.4.

6.3 Particle Swam Optimization (PSO)

The Particle Swam Optimization (PSO) is a computational technique that optimizes a given problem through iterative update on some solution candidates (Parsopoulos & Vrahatis, 2002). The PSO was first introduced by Russel C. Eberhart and James Kennedy in 1995. It is an adaptive and swarm intelligence meta-heuristic algorithm based on socio-psycological paradigm. Specifically, the develoment PSO was based on observation of a group of animal behaviors such as bird flocks or fish schools. Just like the GA, PSO is a population-based method, since it represents the state of the algorithm by a population, which is modified iteratively until a stopping criteria is met. Herein, a population of individuals (solution candidates or particles) adapts by randomly going back towards previously successful regions (see Figure 6.2). It is to be noted that unlike the GA, PSO do not modify the population from generation to generation, but instead keep the same population, iteratively updating the positions of the members of the population. The PSO has two main functionals (or operators) viz:

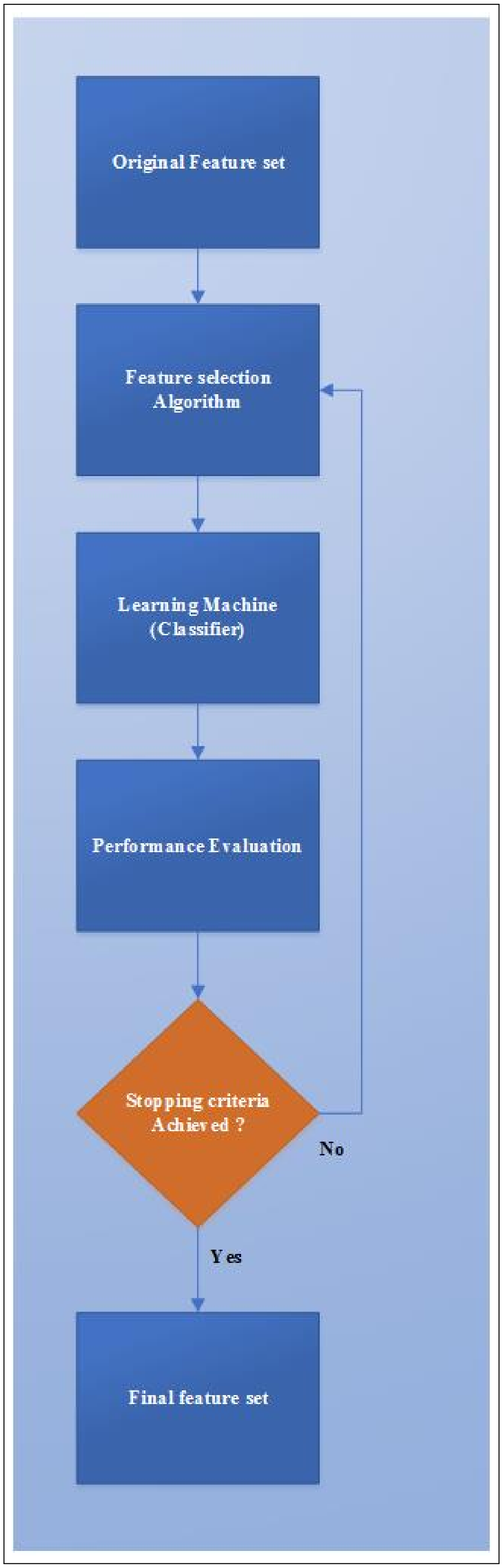

Original Feature set
Feature selection Algorithm
Learning Machine (Classifier)
Performance Evaluation
Stopping criteria Achieved ?
No
Yes
Final feature set

Figure 6.1: Illustrative diagram on generic wrapper-based algorithm

(i) velocity update and

(ii) position update.

During each generation each particle is accelerated towards the particles previous position, and the distance from the global best position. The new velocity is then used to calculate the next position of the particle in the search space. The process is iteratively repeated until some stopping criterias are met. Such may be minimum error. Let $f_{PSO}: \mathbb{R}^n \mapsto \mathbb{R}$ be the cost function associated with the PSO-based feature selection in this work. In other words, the features in the Table 5.2 or are to be reduced to subfeatures that minimizes classification error of our system. In this case the classification output will be the least error associated with the any of the sub-features generated by the binary PSO used herein. Thus the out, say $\boldsymbol{psoERROR} \in \mathbb{R}$, is a single scalar (i.e the scalar 0.0009364 in Figure 6.2). The goal is to find a solution $x^* \in \mathbb{R}$ such that $f_{PSO}(x^*) \leq f_{PSO}(x)$ for all x (different combinatorial set of features from the original dataset) in the search space.

1. Generate a population of agents (also called particles) over a uniform distribution space.

2. Using a suitable objective function, evaluate each particle's position. The fitness function used for the PSO herein is the same as that used for the GA (see Algorithm 4).

3. If the current position of a particle is better than the previous, update it.

4. Determine the best particle according to the particle's previous position.

5. Update the velocities of the particles and the

6. Update the position of the particle

   through the equations 6.3.1 & 6.3.2.

$$V_i^{k+1} = \omega V_i^k + c_1 r_{i1}^k (P_i^k - X_i^k) + c_2 r_{i2}^k (P_{pBEST}^k - X_i^k) \tag{6.3.1}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{6.3.2}$$

Table 6.1: PSO configuration

PSO Parameter	Value
Population size	100
ParticleLength	112
Population type	bitstrings
Fitness Function	kNN-based classification error
Number of Iterations	1000
cognitive parameter c_1	1
social parameter c_2	1
Inertial weight ω	0.25

where

(a) k = iteration number

(b) P_i = positions of the particles.

(c) V_i = velocity (positional change) of the i-th particle.

(d) pBEST = best particle (i.e particle with the best fitness value).

(e) i = 1, 2, 3, ..., N, N = Swarm population size.

(f) ω = inertial weight.

(g) c_1 and c_2 are positive constants called cognitive and social parameters respectively.

(h) r_{i1} and r_{i2} are random numbers uniformly distributed in the range [0, 1].

7. The velocity value is calculated according absolute distance between an individual's data and the target. The further it is, the larger the velocity value.

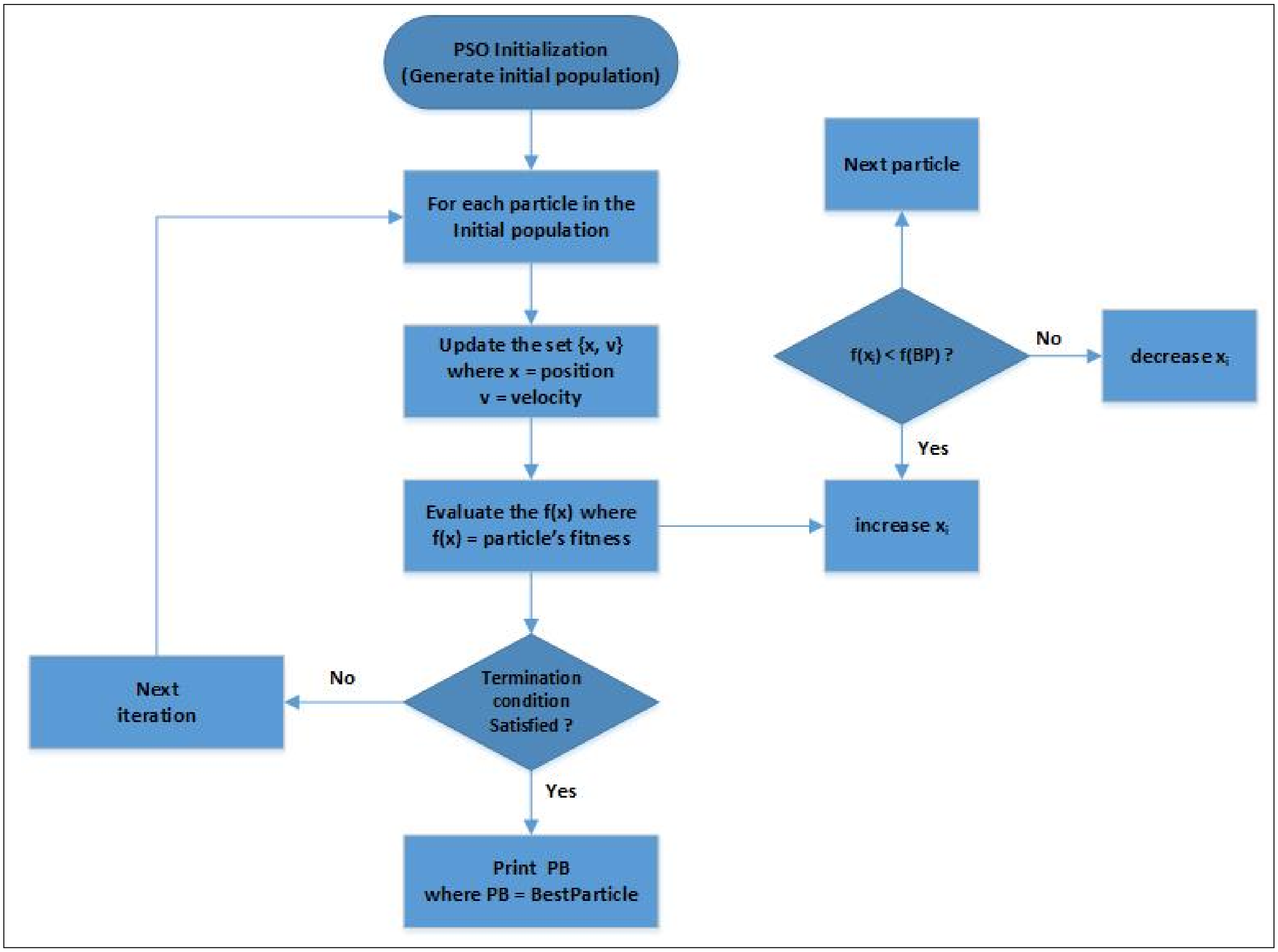

PSO Initialization
(Generate initial population)
For each particle in the Initial population
Update the set {x, v} where x = position v = velocity
Evaluate the f(x) where f(x) = particle's fitness
Termination condition Satisfied ?
No
Next iteration
Yes
Print PB where PB = BestParticle
increase x_i
f(x_i) < f(BP) ?
Yes
No
decrease x_i
Next particle

Figure 6.2: Particle Swarm Optimization (PSO)

The following enumerated points are valid for explaining the underlying PSO used in this work:

1. Original number of features: The candidate solution representing all features (shown in Table 5.1) in the original feature space is by a $\mathbb{R}^{112\times 1}$ matrix given as:

   OriginalString = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1].

   This is also called a particle but this is a special particle as it captures all the features represented in the original feature space. For this special particle, $n = 112$. A single observation from the original dataset using the features associated with these positional indices is given as:

   X_{112} = [0.0579 0.0042 0.0123 0.0034 0.0003 0.1440 0.0980 0.0216 0.0014 0.0293 0.0095 0.0015 0.0001 0.1119 0.0918 0.0468 0.0203 0.0044 0.0004 0.0000 0.0347 0.0449 0.0447 0.0446 0.0444 0.0442 0.0440 0.0439 0.0437 0.0435 0.0345 0.0432 0.0431 0.0429 0.0427 0.0426 0.0424 0.0423 0.0421 0.0420 0.0065 0 -0.0000 0 0.0002 0.0000 0.0002 0.8745 0.0006 0.2469 0.6584 0.3906 0.0015 0.7098 0.5406 0.2331 0.6659 0.6659 0.5964 0.2058 0.5521 0.3833 0.2320 0.0015 0.0109 -0.6364 0.4580 0.6662 0.6664 0.9945 0.9737 0.1000 0.1766 0.0549 0.5226 0.3333 0.9945 0.3565 0.2523 0 1.0000 0.0026 0.0376 0.0052 0.0352 0.0010 0.0155 0.0079 0.0077 0.0031 0.0095 0.0123 0.0013 0.0155 0.0069 0.0230 0.0014 0.1074 0.0062 0.1154 0.0051 0.0047 0.0052 3.0914 3.5708 3.5999 0.3769 0.6231 0.4230 0.0086 0.0078 0.0072]

2. BPSO Particle: A particle in the binary PSO is represented as n-bit string, where n is the number of available features associated with the solution candidate in the feature space. For example if the candidates population is a $\mathit{PopulationSize} \times 112$ matrix of binary strings which can be generated as follows:

   ```
    function PopBits = PopulationFunctionPSO
   ```

```
PopulationSize = 50;

CandidateLength = 112;

RD1 = rand;

PopBits = rand (PopulationSize, CandidateLength) > RD1;

end
```

Any row from ***PopBits*** is a solution candidate. For example suppose a given row is given as :

PSOCandidate = [1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 0 0 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0]

3. Binary digit{0,1}: In each binary string, ”1” represents selected features, while ”0” represents unselected features.

4. How to obtain the features related to a particular candidate: Using the PSOCandidate above, we coding as follows to obtain the associated features:

   PSOCandidate = find (PSOCandidate ==1); which gives the following feature index:

   PSOCandidateFeatureIndex = [1 2 6 8 13 14 15 17 18 20 29 30 34 37 43 44 45 49 53 54 55 57 59 61 64 65 67 68 73 74 75 76 77 79 80 82 83 84 85 86 91 98 101 102 107 109 110]

   The new dataset associated with PSOCandidate will now be given as:

   NewDataset = OldDataset (:, [1 2 6 8 13 14 15 17 18 20 29 30 34 37 43 44 45 49 53 54 55 57 59 61 64 65 67 68 73 74 75 76 77 79 80 82 83 84 85 86 91 98 101 102 107 109 110])

The features generated by PSO are those indexed by the following positional vectors:

$$FeatVect2 = [1, 4, 8, 18, 19, 71, 74, 78, 100, 101, 102, 103, 110, 111]$$

The same number of features were generated by both GA and PSO. This mostly probably, may be due to the same fitness function used in both algorithms. (See Algorithm 4). Out of the 14 features selected by the GA and PSO, 8 were similar. This implies the features common to both are very discriminating. These features were zernike moments, texture properties, and colour features. The diagram in Figure 6.4 shows the features extracted from a single leaf obtained from the Flavia dataset. Some of the numbers shown in both GA-based and PSO-based selected features are the same.

These features also satisfy the definitions 5.1.2 & 5.1.3. Those not selected by the PSO are those that satisfy the definition 5.1.4. The evidence that the PSO converged to a steady state after simulation is shown in Figure 6.3. Only outputs of iterations from generation 151 to 170 are shown for visualization purpose since the whole output from generation 1 to 170 cannot fit into the screen. The total number of functions evaluated in the simulation was 18759 while the optimum generation and the average fitness value at which the Algorithm terminated were 170 and 0.0009364.

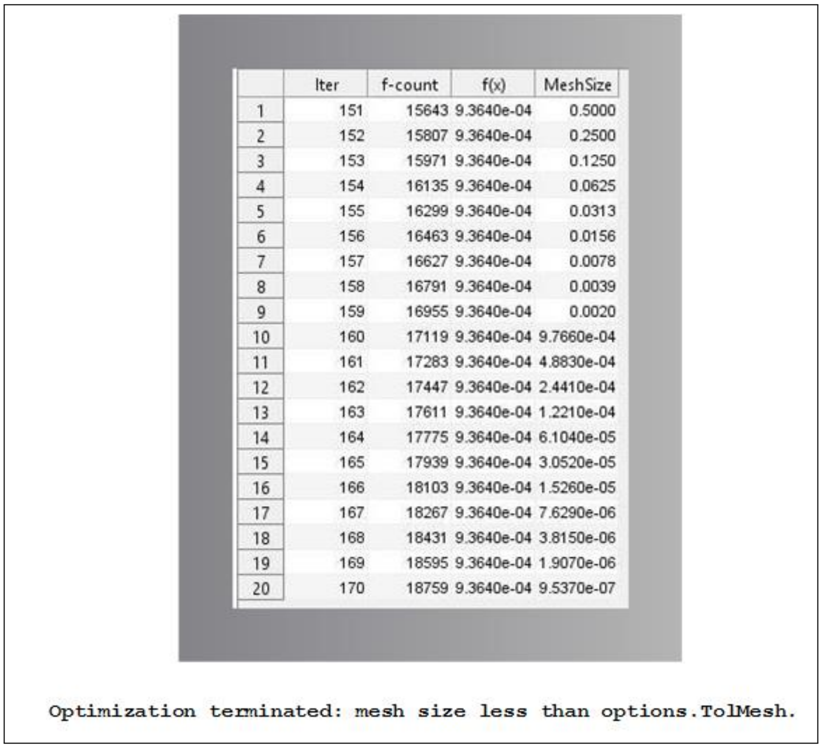
6.4 Analysis of Features

One of the main benefit of feature selection techniques is the elimination of redundancy among the available features in the original feature space. This section provides visualization and analysis of selected features.

6.4.1 Analysis of Variance (ANOVA)

Analysis of Variance (ANOVA) is a statistical technique used to find out whether data from several groups have the same mean (Navidi, 2015).

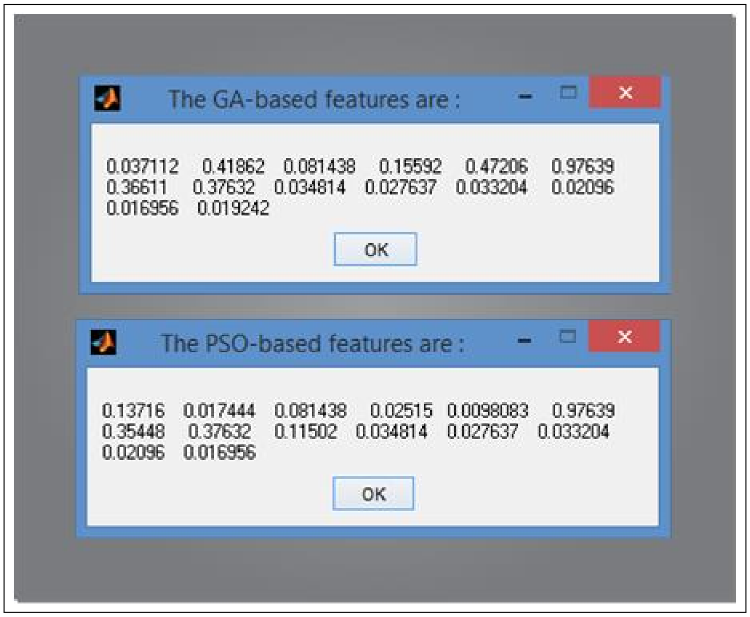
Let the I samples (number of features) in both (GASpace and PSOSpace combined) be represented



	Iter	f-count	f(x)	MeshSize
1	151	15643	9.3640e-04	0.5000
2	152	15807	9.3640e-04	0.2500
3	153	15971	9.3640e-04	0.1250
4	154	16135	9.3640e-04	0.0625
5	155	16299	9.3640e-04	0.0313
6	156	16463	9.3640e-04	0.0156
7	157	16627	9.3640e-04	0.0078
8	158	16791	9.3640e-04	0.0039
9	159	16955	9.3640e-04	0.0020
10	160	17119	9.3640e-04	9.7660e-04
11	161	17283	9.3640e-04	4.8830e-04
12	162	17447	9.3640e-04	2.4410e-04
13	163	17611	9.3640e-04	1.2210e-04
14	164	17775	9.3640e-04	6.1040e-05
15	165	17939	9.3640e-04	3.0520e-05
16	166	18103	9.3640e-04	1.5260e-05
17	167	18267	9.3640e-04	7.6290e-06
18	168	18431	9.3640e-04	3.8150e-06
19	169	18595	9.3640e-04	1.9070e-06
20	170	18759	9.3640e-04	9.5370e-07

```
Optimization terminated: mesh size less than options.TolMesh.
```

Figure 6.3: Evidence of PSO convergence after simulation



The GA-based features are :

```
0.037112   0.41862  0.081438   0.15592   0.47206    0.97639
0.36611   0.37632  0.034814  0.027637  0.033204    0.02096
0.016956   0.019242
```

The PSO-based features are :

```
0.13716  0.017444   0.081438   0.02515  0.0098083    0.97639
0.35448   0.37632   0.11502  0.034814   0.027637   0.033204
0.02096  0.016956
```

Figure 6.4: A sample outputs based on GA-based and PSO-based features for an image

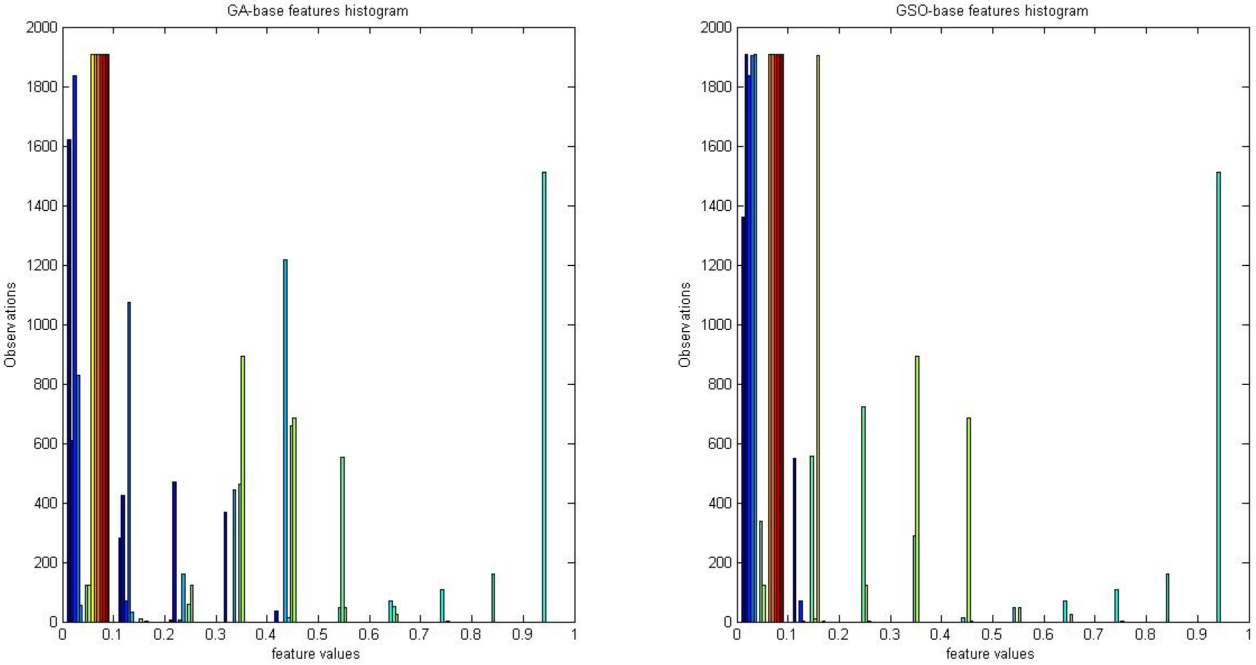
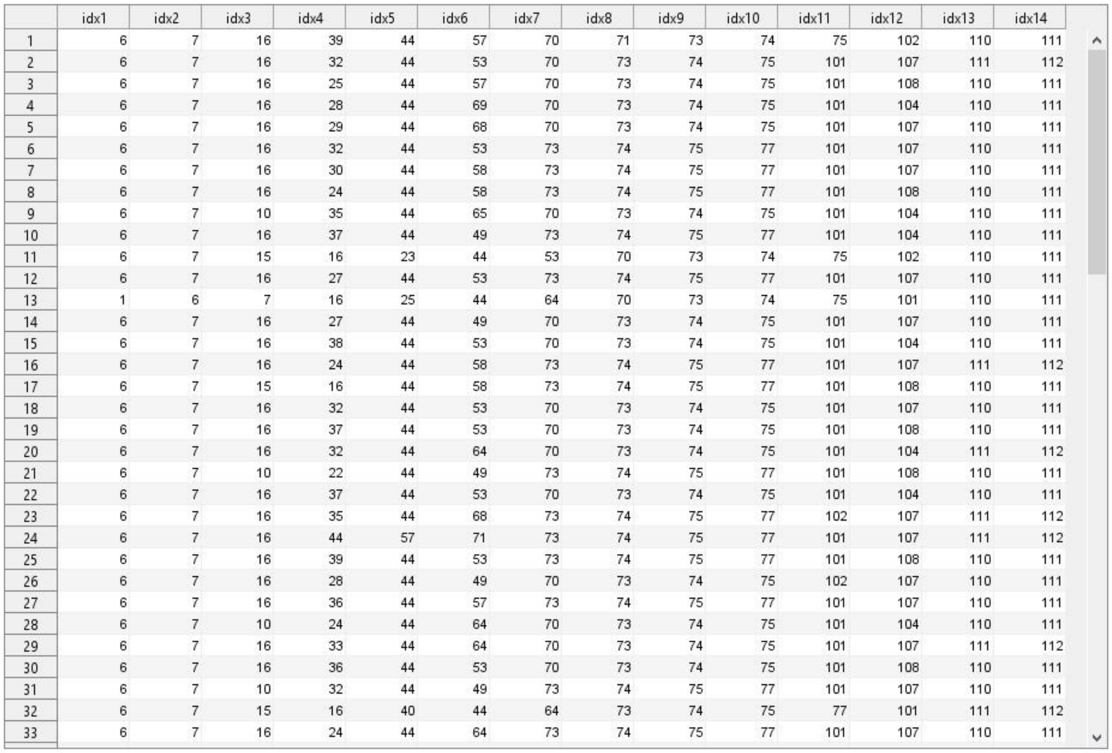


Figure 6.5: Histogram of both GA-based and PSO-based features



	idx1	idx2	idx3	idx4	idx5	idx6	idx7	idx8	idx9	idx10	idx11	idx12	idx13	idx14
1	6	7	16	39	44	57	70	71	73	74	75	102	110	111
2	6	7	16	32	44	53	70	73	74	75	101	107	111	112
3	6	7	16	25	44	57	70	73	74	75	101	108	110	111
4	6	7	16	28	44	69	70	73	74	75	101	104	110	111
5	6	7	16	29	44	68	70	73	74	75	101	107	110	111
6	6	7	16	32	44	53	73	74	75	77	101	107	110	111
7	6	7	16	30	44	58	73	74	75	77	101	107	110	111
8	6	7	16	24	44	58	73	74	75	77	101	108	110	111
9	6	7	10	35	44	65	70	73	74	75	101	104	110	111
10	6	7	16	37	44	49	73	74	75	77	101	104	110	111
11	6	7	15	16	23	44	53	70	73	74	75	102	110	111
12	6	7	16	27	44	53	73	74	75	77	101	107	110	111
13	1	6	7	16	25	44	64	70	73	74	75	101	110	111
14	6	7	16	27	44	49	70	73	74	75	101	107	110	111
15	6	7	16	38	44	53	70	73	74	75	101	104	110	111
16	6	7	16	24	44	58	73	74	75	77	101	107	111	112
17	6	7	15	16	44	58	73	74	75	77	101	108	110	111
18	6	7	16	32	44	53	70	73	74	75	101	107	110	111
19	6	7	16	37	44	53	70	73	74	75	101	108	110	111
20	6	7	16	32	44	64	70	73	74	75	101	104	111	112
21	6	7	10	22	44	49	73	74	75	77	101	108	110	111
22	6	7	16	37	44	53	70	73	74	75	101	104	110	111
23	6	7	16	35	44	68	73	74	75	77	102	107	111	112
24	6	7	16	44	57	71	73	74	75	77	101	107	111	112
25	6	7	16	39	44	53	73	74	75	77	101	108	110	111
26	6	7	16	28	44	49	70	73	74	75	102	107	110	111
27	6	7	16	36	44	57	73	74	75	77	101	107	110	111
28	6	7	10	24	44	64	70	73	74	75	101	104	110	111
29	6	7	16	33	44	64	70	73	74	75	101	107	111	112
30	6	7	16	36	44	53	70	73	74	75	101	108	110	111
31	6	7	10	32	44	49	73	74	75	77	101	107	110	111
32	6	7	15	16	40	44	64	73	74	75	77	101	111	112
33	6	7	16	24	44	64	73	74	75	77	101	107	110	111

Figure 6.6: Relative frequencies of selected features using Mutual Information as the GA fitness function. The modal features from 100 GA iterations confirmed the features selected by both GA-kNN and PSO-kNN feature selectors

as $K_1, K_2, K_3, ...K_{28}$ and their total number as $N = \bigcup_{i=1}^{\infty} K_i$. Let also, $X \;=\; GASpace \bigcup PSOSpace$ The hypothesis here is to test whether

1. $H_0 : \mu_1 = \mu_2, ...\mu_{28}$

   versus

2. H_1 : two or more $\mu_i, i = 1(1)28$, are different.

Let X_{ij} be the jth observation in the ith sample from X. The sample mean of the ith is represented

as:

$$\bar{X_i} = \frac{\sum_{k=1}^{K_i} X_{ij}}{K_i} \tag{6.4.1}$$

The sample grand mean, the average of all the sampled items in X taken together, is given as:

$$\bar{X_{..}} = \frac{\sum_{i=1}^{I} \sum_{k=1}^{K_i} X_{ij}}{N} = \frac{\sum_{i=1}^{I} K_i \bar{X_i}}{N} \tag{6.4.2}$$

Let

SSTr be the variation of the sample means around the grand mean.

SSE be the variation in the individual sample points around their respective sample means.

MSTr be the treatment mean square

MSE be the error mean square

F be the test statistics

The algorithm for the ANOVA can now be given as:

6.4.1.1 Analysis of variance on GA-based features

ANOVA was used to test the means of the features selected by both GA and PSO. The rational for the application of ANOVA techniques is to ascertain that there is no redundancy in the features selected by the GA. An appropriate ANOVA table has columns containing the following items:

Algorithm 5 Computation of ANOVA

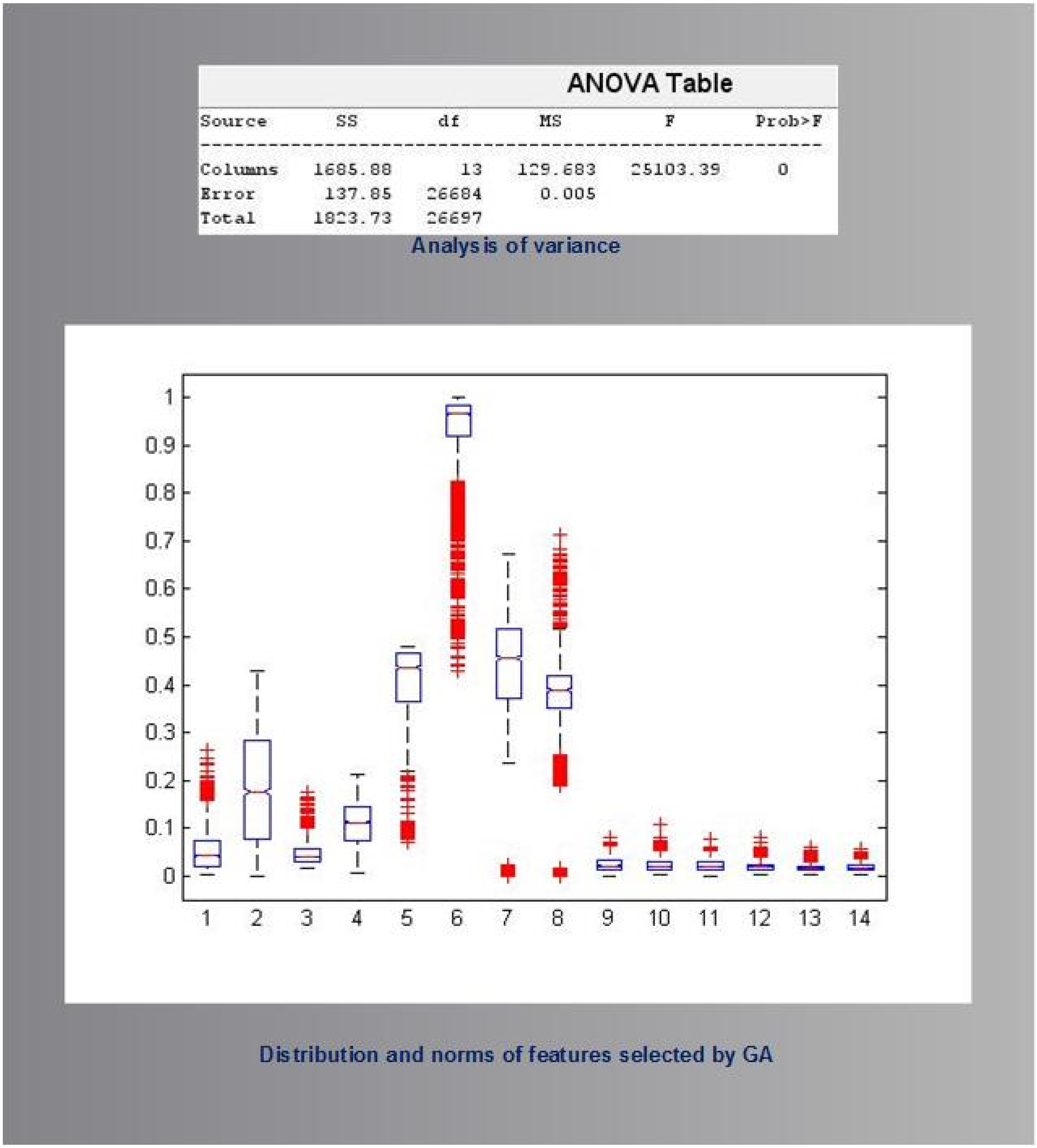
```
procedure ComputeANOVA()
    Input $X$
    To Test $H_0 : \mu_1 = \mu_2, ...\mu_{28}$ versus $H_1$ : two or more $\mu_i, i = 1(1)28$, are different.
    Compute SSTr = $\sum_{i=1}^{I} K_i(\bar{X_{i.}} - \bar{X..})^2$
    Compute SSE = $\sum_{i=1}^{I} \sum_{k=1}^{K_i} (X_{ij} - X_{i.})^2$.
    Compute MSTr = $\frac{SSTr}{I-1}$ and MSE $\frac{SSE}{N-1}$

    Compute $F = \frac{MSTr}{MSE}$.
    Look for: the p-value by looking at the F-Table with I-1 and N-1 degrees of freedom.
end procedure
```

sum of squares (SS), degrees of freedom (df), mean squares (MS), F-statistics and p-value. The p-value is the most important metric from any ANOVA output table. The p-value returned by the ANOVA used for GA-based features was 0 which is less than 0.05. The null hypothesis is rejected and thus, the inference here is that the data from the 14 features selected by the GA do not have the same mean, and hence the GA-based features were not redundant. The SS for columns, error and total were 1685.88, 137.85 and 1823.73. The degrees of freedom for columns, error, and the total were 13, 26684 and 26697 while the mean square and F-statistics were respectively 25103.39 and 0.005 (see Figure 6.7). Thus the p-values was 0, and this shows the effectiveness of the feature selection techniques developed in this study.

6.4.1.2 Analysis of variance on PSO-based features

Similar to section 6.4.1.1 ANOVA was applied on the PSO-based features too. The output is shown in Figure 6.8. Again, the p-value is 0. The SS for columns, error and total were 1516.66, 72.08 and 1388.73. The degrees of freedom for columns, error, and the total were 13, 26604 and 26697 while the mean square and F-statistics for the source (i.e columns, error and total) were respectively 116.67 and 0.003. The F-statistics was 43192.18. The p-value depends fully on other items in the ANOVA table. So, p-values are sufficient to make conclusion about any hypothesis being tested. Thus the PSO-based features used were not redundant. They have different means.



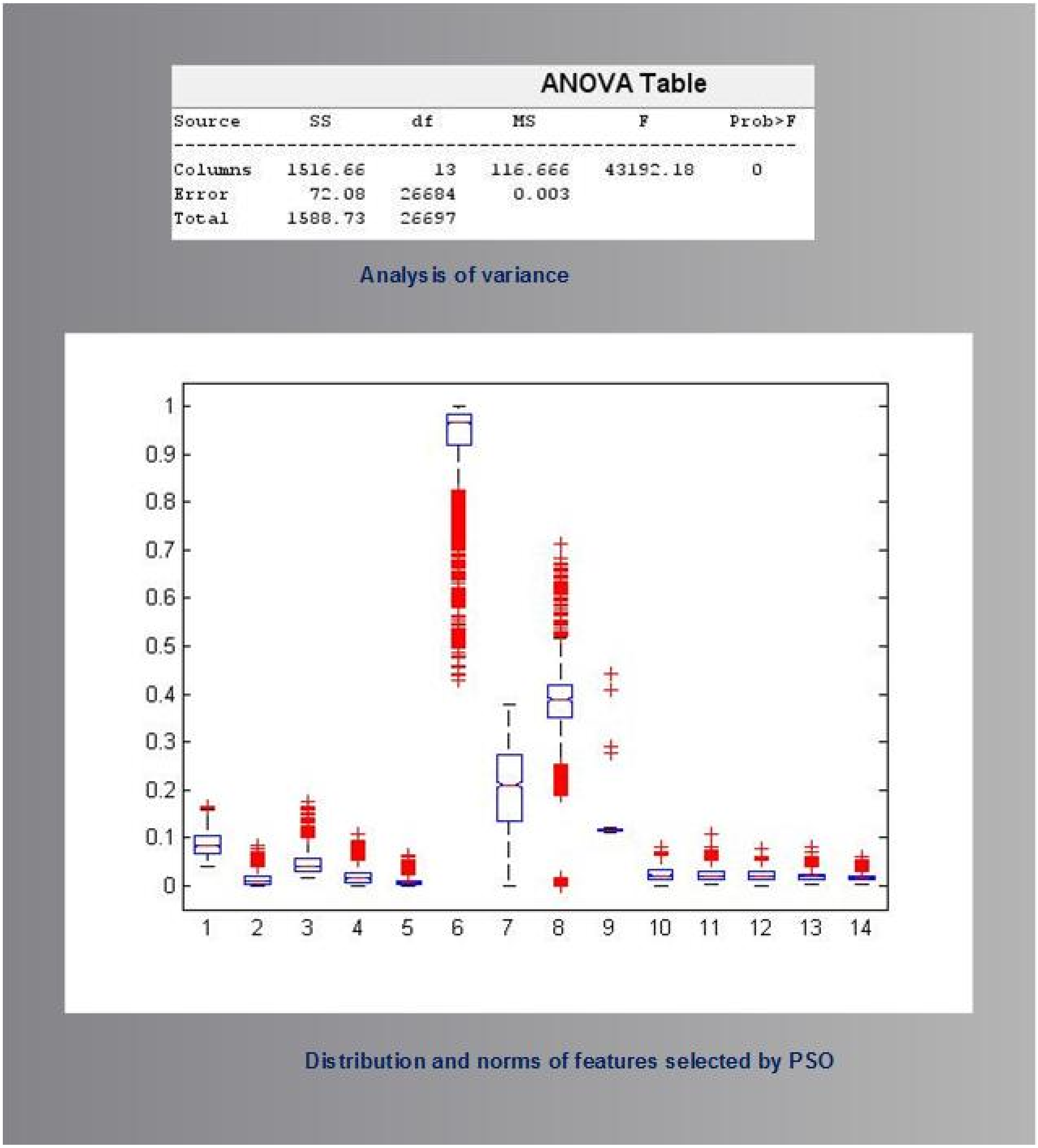
ANOVA Table

```
Source        SS        df        MS          F       Prob>F
-------------------------------------------------------------
Columns    1685.88       13    129.683    25103.39      0
Error       137.85    26684      0.005
Total      1823.73    26697
```

Analysis of variance

Distribution and norms of features selected by GA

Figure 6.7: Analysis of Variance and norms distribution for GA-based features



ANOVA Table

```
Source        SS        df        MS          F        Prob>F
-------------------------------------------------------------
Columns    1516.66       13    116.666    43192.18       0
Error        72.08    26684      0.003
Total      1588.73    26697
```

Analysis of variance

Distribution and norms of features selected by PSO

Figure 6.8: Analysis of Variance and norms distribution for PSO-based features

6.4.2 Correlation coefficients

Correlation coefficient (CC) is a statistical measure of the degree or strength of association between two variables. Given variables $\{x_i, y_i\}, i = 1(1)N$, the CC is defined as

$$CC = r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6.4.3}$$

The r is real number that is trapped between -1 and +1. A value of 1 implies that a linear equation describes the relationship between x and y perfectly, with all data points lying on a line for which y increases as x increases. A value of −1 implies that all data points lie on a line for which y decreases as x increases. A value of 0 implies that there is no linear correlation between the variables. Interestingly, the dimension of both GA-based and PSO-based features were 1907×14, and so, a correlation technique was applied on both GA-based and PSO-based features as $CC = r = COMPUTECORRELATION(GASpace, PSOSpace)$ to generate 14×14 real matrix containing the pairwise correlation coefficient between each pair of columns in the GASpace and PSOSpace. The average correlation coefficients between the GA-based and PSO-based features was {0.9153 0.8697 0.8935 0.8874 0.8352 0.7649 0.9213 0.8410 0.3428 0.9298 0.9152 0.9351 0.9186 0.9088 }, while the overall mean coefficient was 0.8485. These coefficients show high level of correlation between the GA-based and PSO-based features as their values are close to 1 (see Figure 6.10). A rounded version (nearest whole number) of these coefficients is shown in the same figure (named as rounded correlation coefficients). Only the features associated with column 9 (when the features are sorted seems) uncorrelated. The rest of the entries in the binary output are 1s except only one zero in the position {6,5}. The p-values associated with the correlation coefficients are shown in Figure 6.11 as 14×14 matrix of real numbers. The entries in this matrix are results of testing the null hypothesis of no correlation against the alternative that there is a non-zero correlation. Each element of the matrix is the p-value for the corresponding element of matrix in Figure 6.10. If the p-value at the point $\{i, j\}$ is small, say less than 0.05, then the assoaciated correlation coefficient $\rho(i, j)$ is significantly different from zero (see Equation 6.4.4).

$$NullHypotheis = \begin{cases} Accepted & \text{if} \quad \text{pvalue} \quad > \quad 0.05 \\ Rejected & \text{if} \quad \text{pvalue} \quad \leq \quad 0.05 \end{cases} \tag{6.4.4}$$

This implies that the null hypothesis should be rejected. Detailed descriptions of the correlation and statistical methods used in this section can be found in (Kendall, 1970; Best & Roberts, 1975; Gibbons & Chakraborti, 2011; Hollander, Wolfe, & Chicken, 2013; Mathsworks, 2013; Mathworks, 2015; Navidi, 2015).

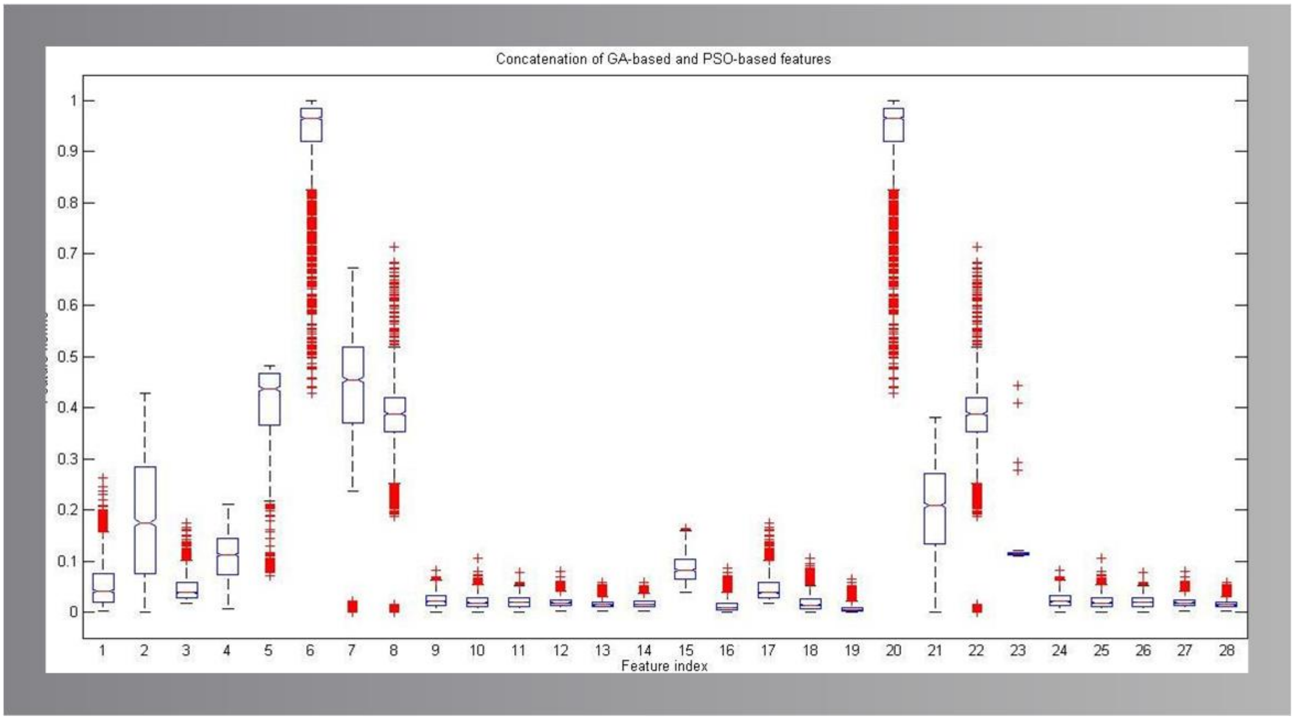
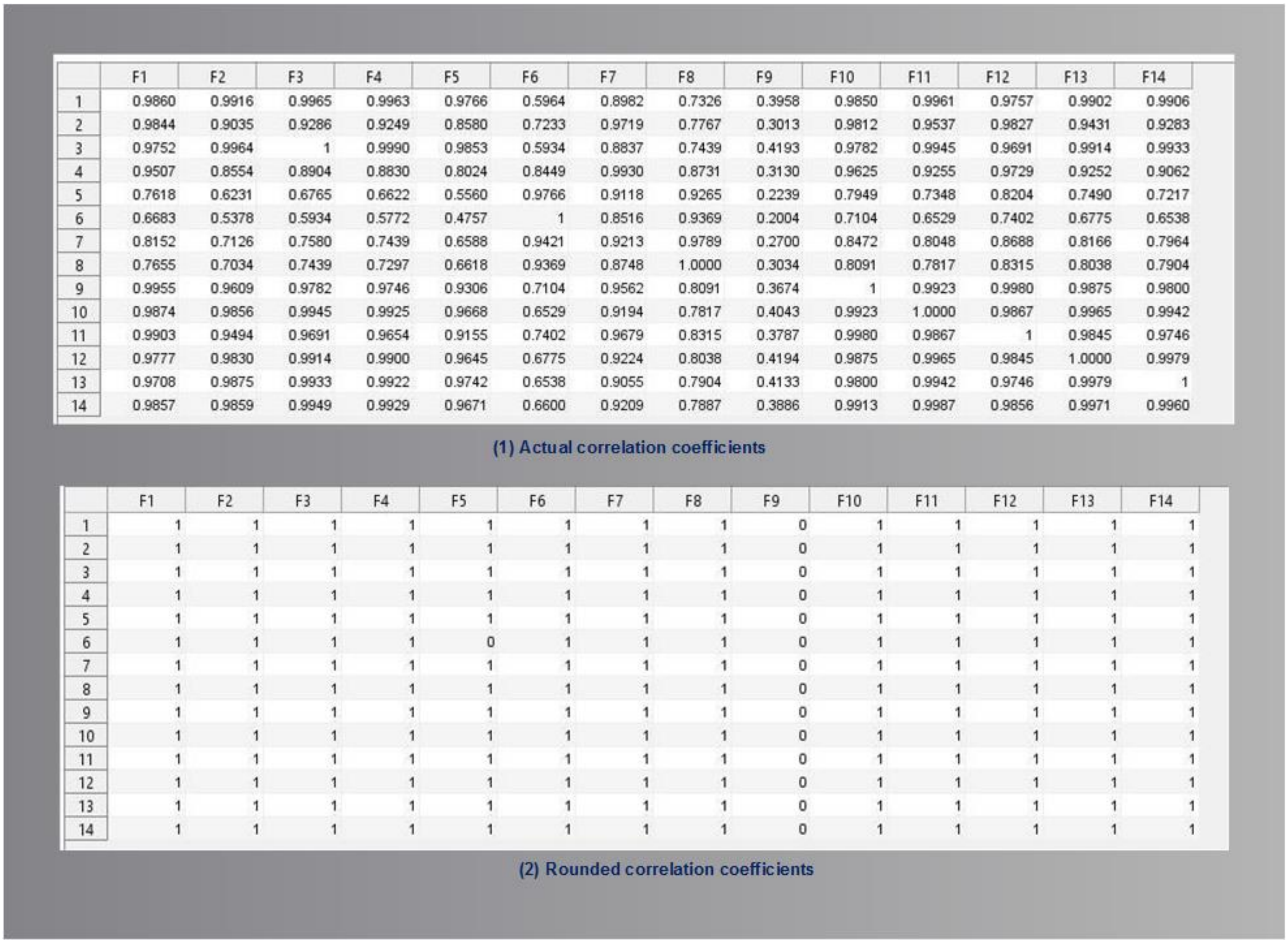


Figure 6.9: Distribution and norms of concatenated PSO-based and GA-based features



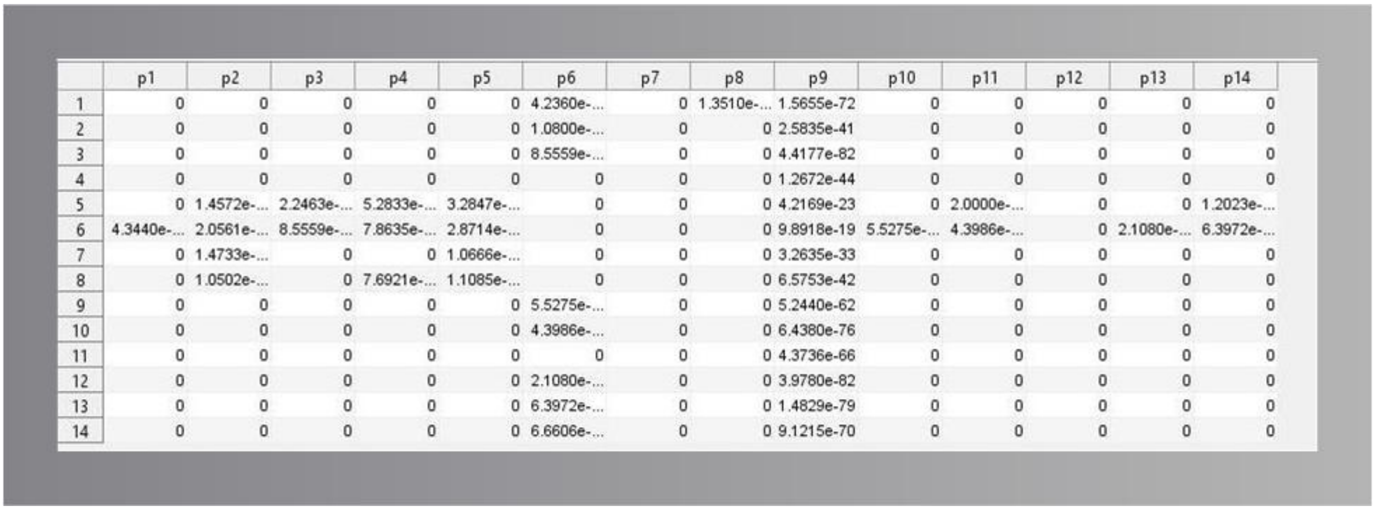
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
1	0.9860	0.9916	0.9965	0.9963	0.9766	0.5964	0.8982	0.7326	0.3958	0.9850	0.9961	0.9757	0.9902	0.9906
2	0.9844	0.9035	0.9286	0.9249	0.8580	0.7233	0.9719	0.7767	0.3013	0.9812	0.9537	0.9827	0.9431	0.9283
3	0.9752	0.9964	1	0.9990	0.9853	0.5934	0.8837	0.7439	0.4193	0.9782	0.9945	0.9691	0.9914	0.9933
4	0.9507	0.8554	0.8904	0.8830	0.8024	0.8449	0.9930	0.8731	0.3130	0.9625	0.9255	0.9729	0.9252	0.9062
5	0.7618	0.6231	0.6765	0.6622	0.5560	0.9766	0.9118	0.9265	0.2239	0.7949	0.7348	0.8204	0.7490	0.7217
6	0.6683	0.5378	0.5934	0.5772	0.4757	1	0.8516	0.9369	0.2004	0.7104	0.6529	0.7402	0.6775	0.6538
7	0.8152	0.7126	0.7580	0.7439	0.6588	0.9421	0.9213	0.9789	0.2700	0.8472	0.8048	0.8688	0.8166	0.7964
8	0.7655	0.7034	0.7439	0.7297	0.6618	0.9369	0.8748	1.0000	0.3034	0.8091	0.7817	0.8315	0.8038	0.7904
9	0.9955	0.9609	0.9782	0.9746	0.9306	0.7104	0.9562	0.8091	0.3674	1	0.9923	0.9980	0.9875	0.9800
10	0.9874	0.9856	0.9945	0.9925	0.9668	0.6529	0.9194	0.7817	0.4043	0.9923	1.0000	0.9867	0.9965	0.9942
11	0.9903	0.9494	0.9691	0.9654	0.9155	0.7402	0.9679	0.8315	0.3787	0.9980	0.9867	1	0.9845	0.9746
12	0.9777	0.9830	0.9914	0.9900	0.9645	0.6775	0.9224	0.8038	0.4194	0.9875	0.9965	0.9845	1.0000	0.9979
13	0.9708	0.9875	0.9933	0.9922	0.9742	0.6538	0.9055	0.7904	0.4133	0.9800	0.9942	0.9746	0.9979	1
14	0.9857	0.9859	0.9949	0.9929	0.9671	0.6600	0.9209	0.7887	0.3886	0.9913	0.9987	0.9856	0.9971	0.9960

(1) Actual correlation coefficients

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
2	1	1	1	1	1	1	1	1	0	1	1	1	1	1
3	1	1	1	1	1	1	1	1	0	1	1	1	1	1
4	1	1	1	1	1	1	1	1	0	1	1	1	1	1
5	1	1	1	1	1	1	1	1	0	1	1	1	1	1
6	1	1	1	1	0	1	1	1	0	1	1	1	1	1
7	1	1	1	1	1	1	1	1	0	1	1	1	1	1
8	1	1	1	1	1	1	1	1	0	1	1	1	1	1
9	1	1	1	1	1	1	1	1	0	1	1	1	1	1
10	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	1	1	1	1	1	1	1	0	1	1	1	1	1
12	1	1	1	1	1	1	1	1	0	1	1	1	1	1
13	1	1	1	1	1	1	1	1	0	1	1	1	1	1
14	1	1	1	1	1	1	1	1	0	1	1	1	1	1

(2) Rounded correlation coefficients

Figure 6.10: Correlation coefficients between the PSO-based and GA-based features



	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14
1	0	0	0	0	0	4.2360e-...	0	1.3510e-...	1.5655e-72	0	0	0	0	0
2	0	0	0	0	0	1.0800e-...	0	0	2.5835e-41	0	0	0	0	0
3	0	0	0	0	0	8.5559e-...	0	0	4.4177e-82	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1.2672e-44	0	0	0	0	0
5	0	1.4572e-...	2.2463e-...	5.2833e-...	3.2847e-...	0	0	0	4.2169e-23	0	2.0000e-...	0	0	1.2023e-...
6	4.3440e-...	2.0561e-...	8.5559e-...	7.8635e-...	2.8714e-...	0	0	0	9.8918e-19	5.5275e-...	4.3986e-...	0	2.1080e-...	6.3972e-...
7	0	1.4733e-...	0	0	1.0666e-...	0	0	0	3.2635e-33	0	0	0	0	0
8	0	1.0502e-...	0	7.6921e-...	1.1085e-...	0	0	0	6.5753e-42	0	0	0	0	0
9	0	0	0	0	0	5.5275e-...	0	0	5.2440e-62	0	0	0	0	0
10	0	0	0	0	0	4.3986e-...	0	0	6.4380e-76	0	0	0	0	0
11	0	0	0	0	0	0	0	0	4.3736e-66	0	0	0	0	0
12	0	0	0	0	0	2.1080e-...	0	0	3.9780e-82	0	0	0	0	0
13	0	0	0	0	0	6.3972e-...	0	0	1.4829e-79	0	0	0	0	0
14	0	0	0	0	0	6.6606e-...	0	0	9.1215e-70	0	0	0	0	0

Figure 6.11: The p-values of all correlation coefficients between GA-based and PSO-based features

6.5 Principal Component Analysis (PCA)

PCA is a mathematical procedure (orthogonal transformation from applied linear algebra) that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called the principal components. PCA is a dimensionality reduction technique and is useful for dimension reduction when the transformed features have a descriptive power more easily ordered than the original features. It is used in selecting a subset of variables from a large dataset, based on which the original variables have the highest correlations with the principal component. In other words, PCA combines the variables linearly such that the maximum variance is extracted from the variables. In terms of geometry, PCA involves the rotation of the axes of the coordinate system of the original variable to new orthogonal axes, commonly called principal axes, such that the new axes aligns with the directions of maximum variation of the original variables (observations). The property of the maximum variation of the projected points defines the first principal axis and it's the line or direction with maximum variation of the projected values of the original data points. These projected values are called principal component scores (Cambell & Atchlev, 1981; Dutta, Hines, Gardner, & Boilot, 2002). It should be noted that each principal component is a linear combination of the original variables and all the principal components are orthogonal to each other, so there is no redundant information. Thus, PCA is both feature transformation and reduction technique. The PCA accepts a dataset and rotates it in such away that the maximum variability is visible. The operation of PCA is given as follows:

1. Given a dataset $\{x_n\}, n = 1(1)N$, and x_n is a N-dimensional vector, (N = 112 for this study).

2. Task?: To project the given data onto an M-dimensional subpace, where $M < N$, and $M, N \in Z^+$.

3. Assumption: The projection is assumed to be represented as

$$y = Ax \tag{6.5.1}$$

where

$$A = [u_1^T, u_2^T, \cdots, u_n^T,]$$

and

$$u^T{}_i u_i = 1 \;\; for \;\; i = 1(1)M \tag{6.5.3}$$

The objective now is to maximize the variance of y_n, which is the trace of the covariance of matrix y_n. Therefore the actual objective now is to find $max(trace(H_y))$ where

$$H_y = \frac{1}{N}\sum_{n=1}^{N}(y_n - \bar{y})^T, \;\; \bar{y} = \frac{1}{N}\sum_{n=1}^{N} y_n \tag{6.5.4}$$

If we assume H_x to be covariance matrix of x_n and since

$$trace(H_y) = trace(AH_xA^T), \tag{6.5.5}$$

the Langragian multiplier and derivatives gives

$$H_x u_i = \lambda_i u_i \tag{6.5.6}$$

Next, x_n can be represented as

$$x_n = \sum_{i=1}^{M}(x_n^T u_i)u_i \tag{6.5.7}$$

where u_i is the largest vector of H_x which corresponds to the ith largest eigenvalue.

The first 33 instances of the first 14 principal component of the feaures derived from the Flavia dataset are given Figure 6.13. A 2D and 3D plot which allows you to visualize the absolute value and sign of each variable's contribution to the first two or three principal components, and how each observation is represented in terms of those components are shown in Figures 6.14 , 6.15 & 6.16. The number of principal components were automatically selected by a GA as shown in Figure 6.12.

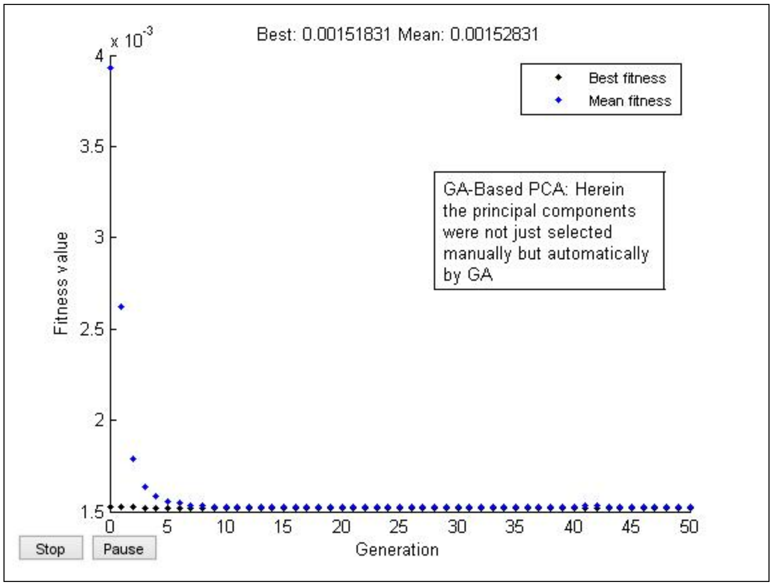
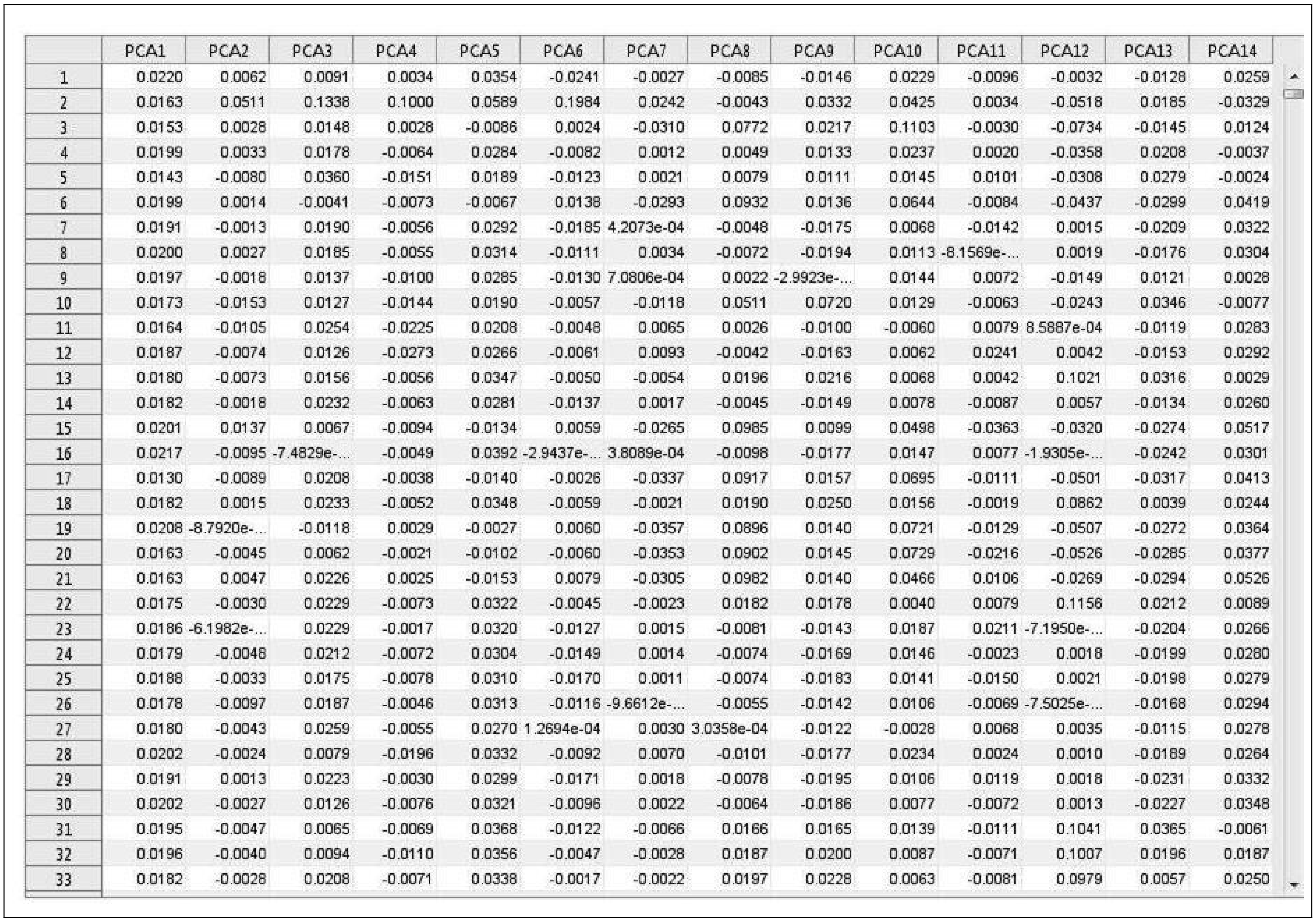


Figure 6.12: Genetic principal components: GA was used to automate the number of principal components (PC) finally used. The final number of PC was 41



	PCA1	PCA2	PCA3	PCA4	PCA5	PCA6	PCA7	PCA8	PCA9	PCA10	PCA11	PCA12	PCA13	PCA14
1	0.0220	0.0062	0.0091	0.0034	0.0354	-0.0241	-0.0027	-0.0085	-0.0146	0.0229	-0.0096	-0.0032	-0.0128	0.0259
2	0.0163	0.0511	0.1338	0.1000	0.0589	0.1984	0.0242	-0.0043	0.0332	0.0425	0.0034	-0.0518	0.0185	-0.0329
3	0.0153	0.0028	0.0148	0.0028	-0.0086	0.0024	-0.0310	0.0772	0.0217	0.1103	-0.0030	-0.0734	-0.0145	0.0124
4	0.0199	0.0033	0.0178	-0.0064	0.0284	-0.0082	0.0012	0.0049	0.0133	0.0237	0.0020	-0.0358	0.0208	-0.0037
5	0.0143	-0.0080	0.0360	-0.0151	0.0189	-0.0123	0.0021	0.0079	0.0111	0.0145	0.0101	-0.0308	0.0279	-0.0024
6	0.0199	0.0014	-0.0041	-0.0073	-0.0067	0.0138	-0.0293	0.0932	0.0136	0.0644	-0.0084	-0.0437	-0.0299	0.0419
7	0.0191	-0.0013	0.0190	-0.0056	0.0292	-0.0185	4.2073e-04	-0.0048	-0.0175	0.0068	-0.0142	0.0015	-0.0209	0.0322
8	0.0200	0.0027	0.0185	-0.0055	0.0314	-0.0111	0.0034	-0.0072	-0.0194	0.0113	-8.1569e-...	0.0019	-0.0176	0.0304
9	0.0197	-0.0018	0.0137	-0.0100	0.0285	-0.0130	7.0806e-04	0.0022	-2.9923e-...	0.0144	0.0072	-0.0149	0.0121	0.0028
10	0.0173	-0.0153	0.0127	-0.0144	0.0190	-0.0057	-0.0118	0.0511	0.0720	0.0129	-0.0063	-0.0243	0.0346	-0.0077
11	0.0164	-0.0105	0.0254	-0.0225	0.0208	-0.0048	0.0065	0.0026	-0.0100	-0.0060	0.0079	8.5887e-04	-0.0119	0.0283
12	0.0187	-0.0074	0.0126	-0.0273	0.0266	-0.0061	0.0093	-0.0042	-0.0163	0.0062	0.0241	0.0042	-0.0153	0.0292
13	0.0180	-0.0073	0.0156	-0.0056	0.0347	-0.0050	-0.0054	0.0196	0.0216	0.0068	0.0042	0.1021	0.0316	0.0029
14	0.0182	-0.0018	0.0232	-0.0063	0.0281	-0.0137	0.0017	-0.0045	-0.0149	0.0078	-0.0087	0.0057	-0.0134	0.0260
15	0.0201	0.0137	0.0067	-0.0094	-0.0134	0.0059	-0.0265	0.0985	0.0099	0.0498	-0.0363	-0.0320	-0.0274	0.0517
16	0.0217	-0.0095	-7.4829e-...	-0.0049	0.0392	-2.9437e-...	3.8089e-04	-0.0098	-0.0177	0.0147	0.0077	-1.9305e-...	-0.0242	0.0301
17	0.0130	-0.0089	0.0208	-0.0038	-0.0140	-0.0026	-0.0337	0.0917	0.0157	0.0695	-0.0111	-0.0501	-0.0317	0.0413
18	0.0182	0.0015	0.0233	-0.0052	0.0348	-0.0059	-0.0021	0.0190	0.0250	0.0156	-0.0019	0.0862	0.0039	0.0244
19	0.0208	-8.7920e-...	-0.0118	0.0029	-0.0027	0.0060	-0.0357	0.0896	0.0140	0.0721	-0.0129	-0.0507	-0.0272	0.0364
20	0.0163	-0.0045	0.0062	-0.0021	-0.0102	-0.0060	-0.0353	0.0902	0.0145	0.0729	-0.0216	-0.0526	-0.0285	0.0377
21	0.0163	0.0047	0.0226	0.0025	-0.0153	0.0079	-0.0305	0.0982	0.0140	0.0466	0.0106	-0.0269	-0.0294	0.0526
22	0.0175	-0.0030	0.0229	-0.0073	0.0322	-0.0045	-0.0023	0.0182	0.0178	0.0040	0.0079	0.1156	0.0212	0.0089
23	0.0186	-6.1982e-...	0.0229	-0.0017	0.0320	-0.0127	0.0015	-0.0081	-0.0143	0.0187	0.0211	-7.1950e-...	-0.0204	0.0266
24	0.0179	-0.0048	0.0212	-0.0072	0.0304	-0.0149	0.0014	-0.0074	-0.0169	0.0146	-0.0023	0.0018	-0.0199	0.0280
25	0.0188	-0.0033	0.0175	-0.0078	0.0310	-0.0170	0.0011	-0.0074	-0.0183	0.0141	-0.0150	0.0021	-0.0198	0.0279
26	0.0178	-0.0097	0.0187	-0.0046	0.0313	-0.0116	-9.6612e-...	-0.0055	-0.0142	0.0106	-0.0069	-7.5025e-...	-0.0168	0.0294
27	0.0180	-0.0043	0.0259	-0.0055	0.0270	1.2694e-04	0.0030	3.0358e-04	-0.0122	-0.0028	0.0068	0.0035	-0.0115	0.0278
28	0.0202	-0.0024	0.0079	-0.0196	0.0332	-0.0092	0.0070	-0.0101	-0.0177	0.0234	0.0024	0.0010	-0.0189	0.0264
29	0.0191	0.0013	0.0223	-0.0030	0.0299	-0.0171	0.0018	-0.0078	-0.0195	0.0106	0.0119	0.0018	-0.0231	0.0332
30	0.0202	-0.0027	0.0126	-0.0076	0.0321	-0.0096	0.0022	-0.0064	-0.0186	0.0077	-0.0072	0.0013	-0.0227	0.0348
31	0.0195	-0.0047	0.0065	-0.0069	0.0368	-0.0122	-0.0066	0.0166	0.0165	0.0139	-0.0111	0.1041	0.0365	-0.0061
32	0.0196	-0.0040	0.0094	-0.0110	0.0356	-0.0047	-0.0028	0.0187	0.0200	0.0087	-0.0071	0.1007	0.0196	0.0187
33	0.0182	-0.0028	0.0208	-0.0071	0.0338	-0.0017	-0.0022	0.0197	0.0228	0.0063	-0.0081	0.0979	0.0057	0.0250

Figure 6.13: The first 14 principal component of the original feature set

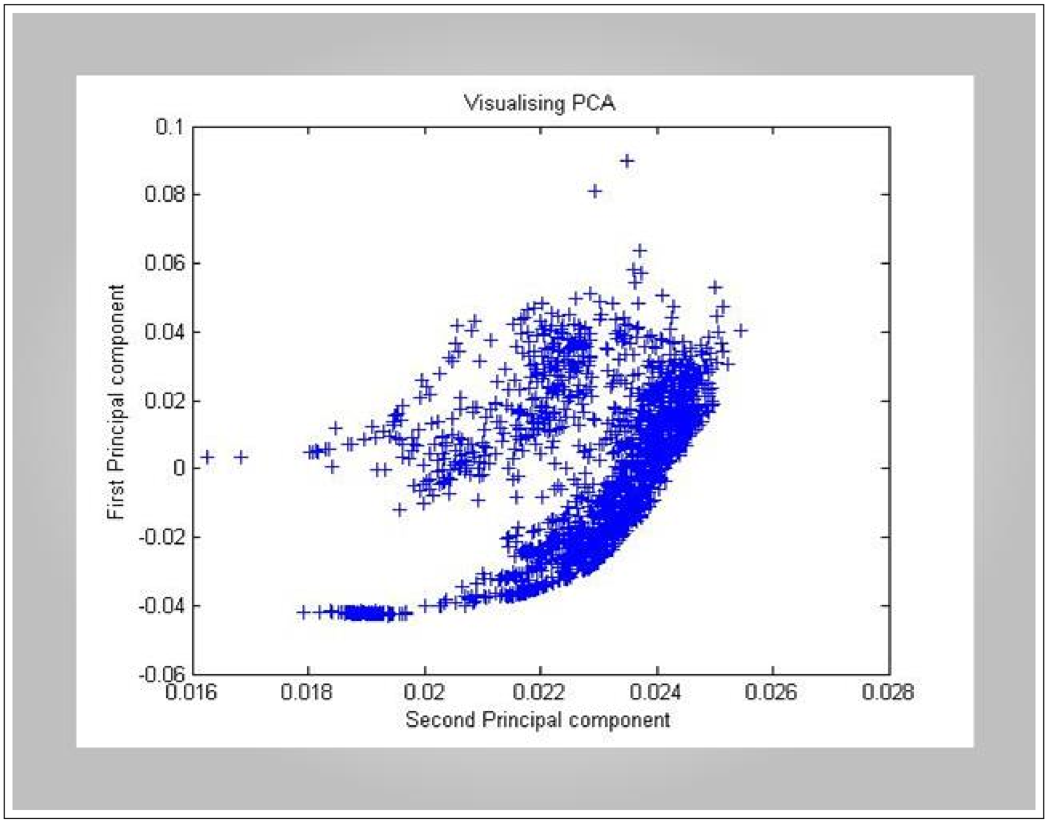


Figure 6.14: Visualization of two PCA axes

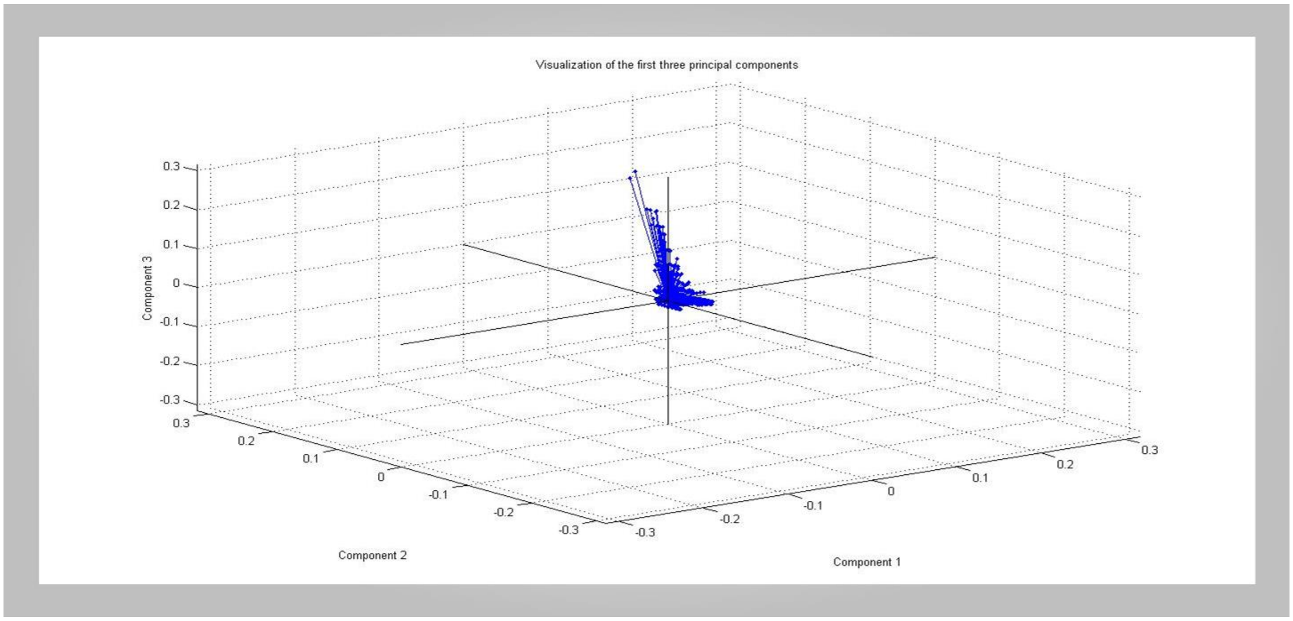


Figure 6.15: A 3D view of the first three principal components

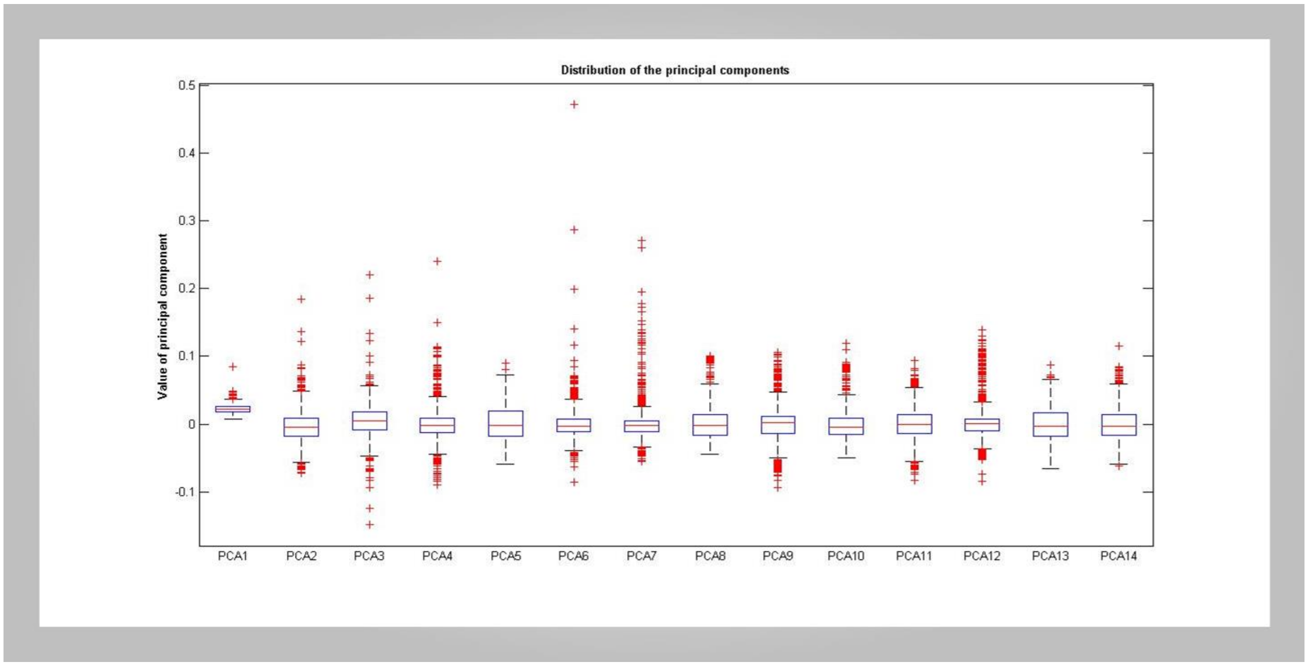


Figure 6.16: Distribution of the principal components

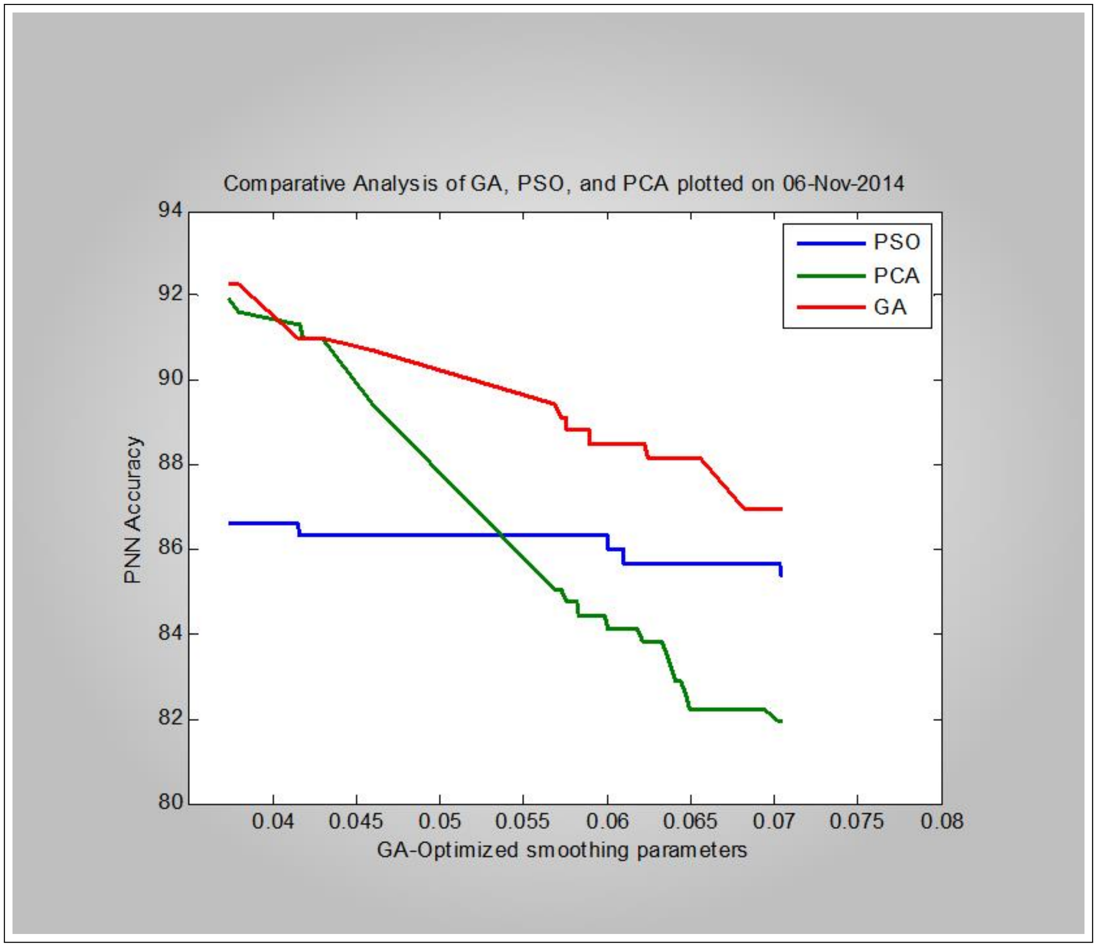


Figure 6.17: Comparative Analysis of GA, PSO, and PCA

6.6 Discussion and Conclusions

This chapter demonstrates the impact of both GA, PSO and PCA on some selected classifiers. The original features space was reduced from a 1907×112 matrix of real numbers to 1907×41 matrix of real numbers using a GA (see Figure 6.12). In other words, both GA and PSO suprisingly selected only 12.50% of the original dataset. The selected features were further analysed statistically using ANOVA and CORRELATION techniques. The results of statistical analysis confirmed the effectiveness of the GA-based and PSO-based feature selection techniques used in that there were no redundant features and there was high degree of correlation between the features selected by both GA and PSO. Nine different classification models were tested as shown in the paper (O. Babatunde, Armstrong, Leng, & Diepeveen, 2015c) published from this thesis.

The results in this section showed that both GA and PSO-based features outperformed the original features while GA-based feature in turn outperformed the PSO-based features. The features selected by both GA and PSO are somewhat similar as shown in Figure 6.5. This may be due to the same fitness function used for both. However the different computational nature of the two evolutionary algorithms (GA and PSO) may be the reason why the number of features selected both algorithms are not entirely the same. Only 57% features were similarly by both algorithms. An indication here is that both PSO and GA are good candidates for feature selections and their application in precision agriculture (computer-based vision systems for automatic identifications of plant species) proved useful as they were able to improve the classification accuracy of the underlying classifiers.

In addition to both GA and PSO, 14 principal components were also extracted from the original feature set since the cardinality of features selected by both GA and PSO were 14. The performances of the three (3) manifold projection techniques (feature reduction algorithms) were further tested on PNN as shown in Figure 6.17 with GA-based algorithm outperforming both PSO and PCA.

Chapter 7

PNN-Based Classifier for Plant Leaves

7.1 Introduction

In pattern recognition or image classification problems, Artificial Neural Networks (ANN) are used to classify given inputs (with unknown class) into a set of target categories. The classification is done by looking for a functional relationship between the class information and the features or attributes in the given dataset. ANN are mathematical abstractions and / or models of Human cognitive system. Among the available neural networks is the Probabilistic Neural Networks (PNN) which is a widely known classifier model for supervised classification (Georgiou, Malefaki, Alevizos, & Vrahatis, 2006). PNN are also closely related to the well-known discrimant analysis since they both use kernel functions for estimating the probability density function (pdf(7.5.5)) of each class in the given dataset. PNN have been applied in diverse areas like bioinformatics, medical informatics, and engineering analysis (Huang, 2002). The PNN unlike other neural networks such as Multi-Layer Perceptron (MLP), inherits its basic concepts from the Baye's rule and Bayesian Classifier (Kim, Lee, Lee, & Chang, 2005). Therefore, it will be useful to describe the Bayesian approach to classification before talking about PNN. PNN was applied in this work to classify some set of plant species from the Flavia dataset (Wu et al., 2007) and compared the results from this study with Wu et al. (2007).

7.2 The Flavia Dataset

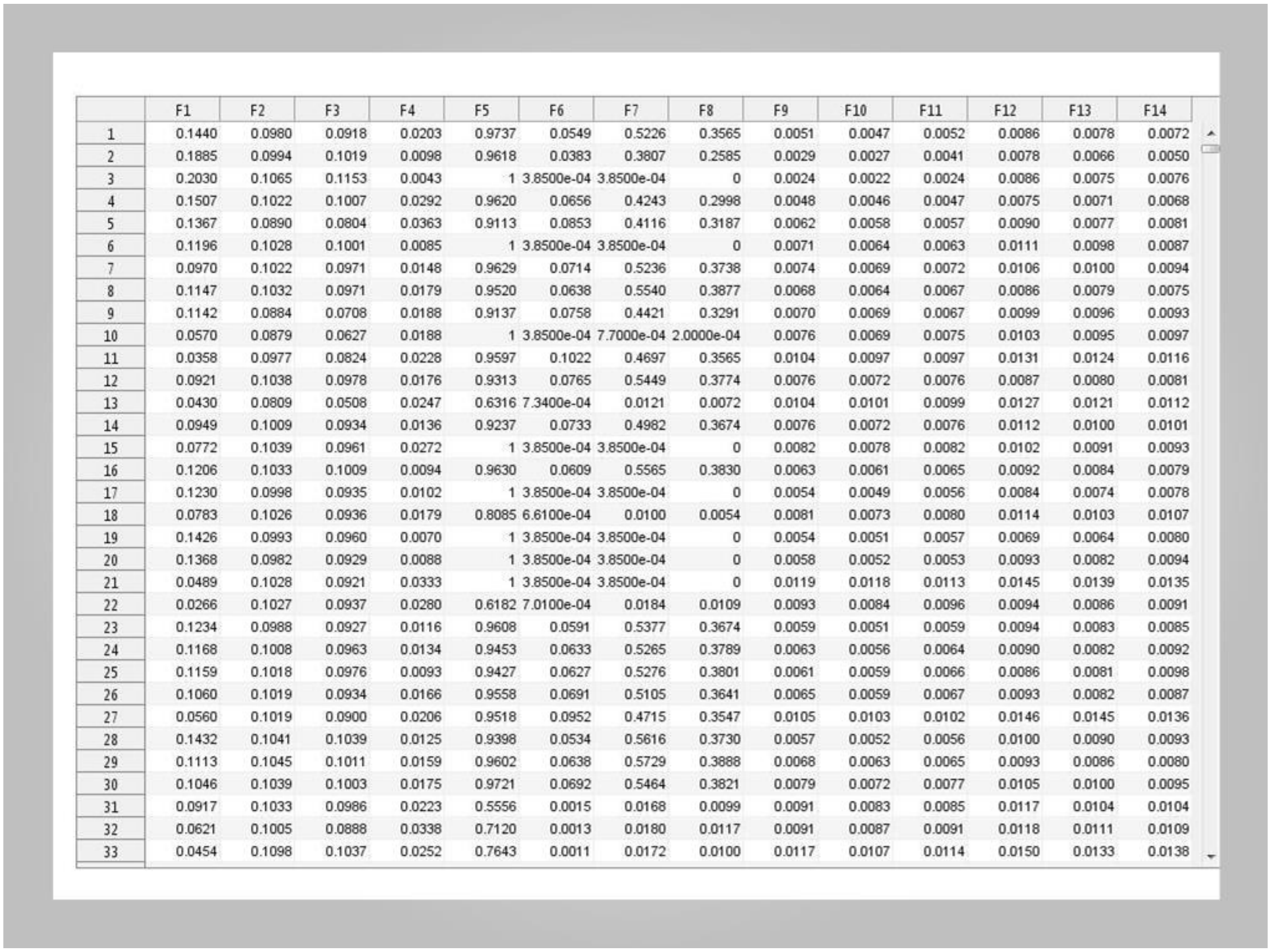
The source of images of leaves used in this study are images of leaves found in the Flavia dataset which is publicly available (Wu et al., 2007). The Flavia dataset is a constrained set of leaf images taken against a white background and without any stem present. The species in the dataset have a varying number of instances (This is shown in Table 7.3). The dataset has 1907 images of 32 species of plants. For this study, the dataset was divided into two disjoint sets, each of which contains 1587 images and 320 images for both training and test set respectively.

7.3 Features Generated From The Flavia Dataset

A total number of 112 features were generated from the Flavia dataset. These features include Zernike Moments (ZM), Fourier Descriptors (FD), Lengendre Moments (LM), Hu 7 Moments (Hu7M), Texture Properties (TP) , Geometrical Properties (GP) and Colour features. A Genetic Algorithm (GA) has been used to reduce the feature space from 100 to 14 (see Figure 7.1). Table 5.1 shows the representation of the features and class information for our dataset.

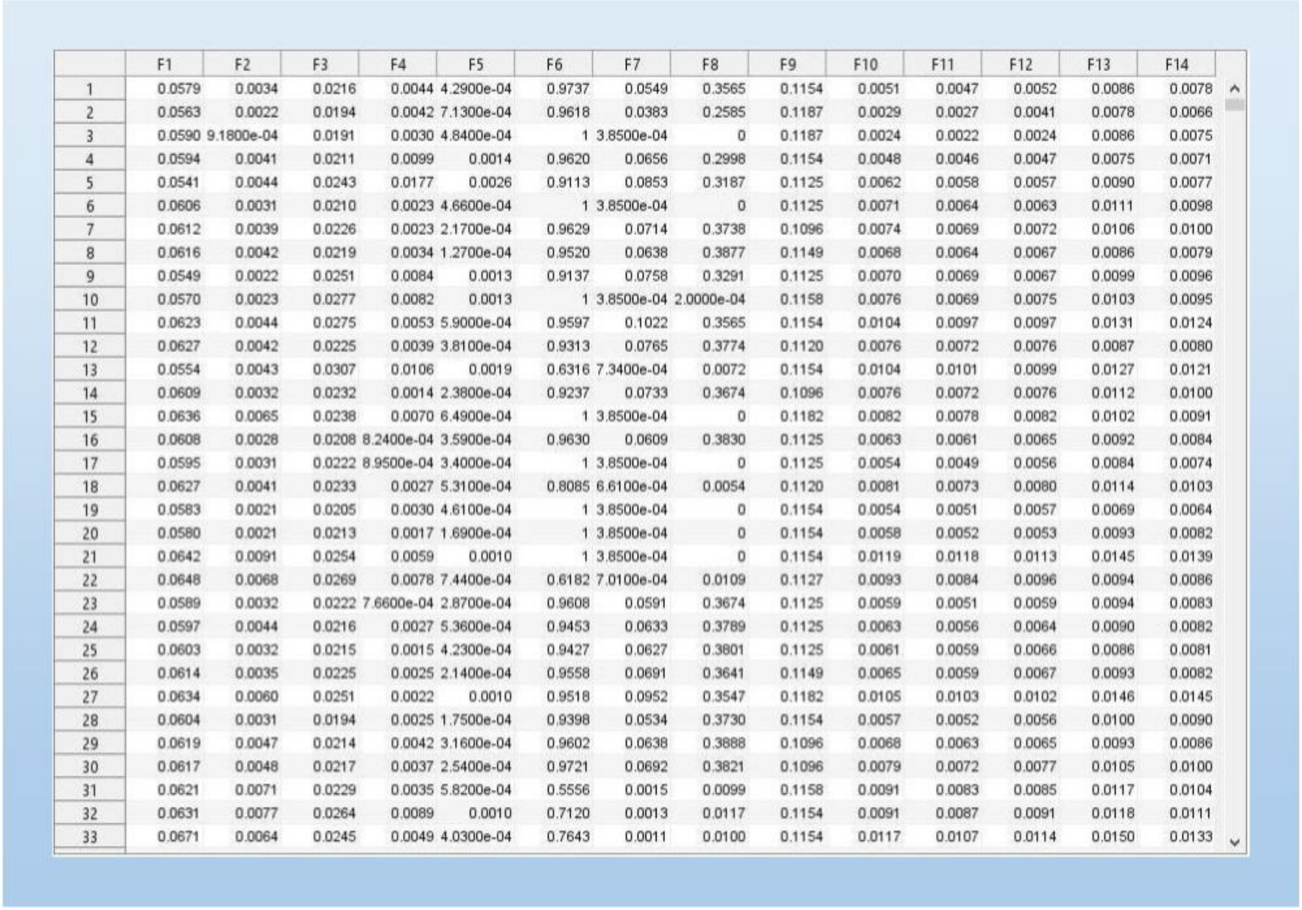
Table 7.1: Table showing 14 features derived from the Flavia Dataset

Observation	F1 F2 F3 F4 F5 ... F14	Class No
Image1	$X_{1,1}$ $X_{1,2}$ $X_{1,3}$ $X_{1,4}$ $X_{1,5}$... $X_{1,14}$	1
Image2	$X_{2,1}$ $X_{2,2}$ $X_{2,3}$ $X_{2,4}$ $X_{2,5}$... $X_{2,14}$	1
Image3	$X_{3,1}$ $X_{3,2}$ $X_{3,3}$ $X_{3,4}$ $X_{3,5}$... $X_{3,14}$	1
Image4	$X_{4,1}$ $X_{4,2}$ $X_{4,3}$ $X_{4,4}$ $X_{4,5}$... $X_{4,14}$	.
Image5	$X_{5,1}$ $X_{5,2}$ $X_{5,3}$ $X_{5,4}$ $X_{5,5}$... $X_{5,14}$	.
Image6	$X_{6,1}$ $X_{6,2}$ $X_{6,3}$ $X_{6,4}$ $X_{6,5}$... $X_{6,14}$	.
...	...	32
...	...	32
...	...	32
Image1907	$X_{1907,1}$ $X_{1907,2}$ $X_{1907,3}$... $X_{1907,14}$	32



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
1	0.1440	0.0980	0.0918	0.0203	0.9737	0.0549	0.5226	0.3565	0.0051	0.0047	0.0052	0.0086	0.0078	0.0072
2	0.1885	0.0994	0.1019	0.0098	0.9618	0.0383	0.3807	0.2585	0.0029	0.0027	0.0041	0.0078	0.0066	0.0050
3	0.2030	0.1065	0.1153	0.0043	1	3.8500e-04	3.8500e-04	0	0.0024	0.0022	0.0024	0.0086	0.0075	0.0076
4	0.1507	0.1022	0.1007	0.0292	0.9620	0.0656	0.4243	0.2998	0.0048	0.0046	0.0047	0.0075	0.0071	0.0068
5	0.1367	0.0890	0.0804	0.0363	0.9113	0.0853	0.4116	0.3187	0.0062	0.0058	0.0057	0.0090	0.0077	0.0081
6	0.1196	0.1028	0.1001	0.0085	1	3.8500e-04	3.8500e-04	0	0.0071	0.0064	0.0063	0.0111	0.0098	0.0087
7	0.0970	0.1022	0.0971	0.0148	0.9629	0.0714	0.5236	0.3738	0.0074	0.0069	0.0072	0.0106	0.0100	0.0094
8	0.1147	0.1032	0.0971	0.0179	0.9520	0.0638	0.5540	0.3877	0.0068	0.0064	0.0067	0.0086	0.0079	0.0075
9	0.1142	0.0884	0.0708	0.0188	0.9137	0.0758	0.4421	0.3291	0.0070	0.0069	0.0067	0.0099	0.0096	0.0093
10	0.0570	0.0879	0.0627	0.0188	1	3.8500e-04	7.7000e-04	2.0000e-04	0.0076	0.0069	0.0075	0.0103	0.0095	0.0097
11	0.0358	0.0977	0.0824	0.0228	0.9597	0.1022	0.4697	0.3565	0.0104	0.0097	0.0097	0.0131	0.0124	0.0116
12	0.0921	0.1038	0.0978	0.0176	0.9313	0.0765	0.5449	0.3774	0.0076	0.0072	0.0076	0.0087	0.0080	0.0081
13	0.0430	0.0809	0.0508	0.0247	0.6316	7.3400e-04	0.0121	0.0072	0.0104	0.0101	0.0099	0.0127	0.0121	0.0112
14	0.0949	0.1009	0.0934	0.0136	0.9237	0.0733	0.4982	0.3674	0.0076	0.0072	0.0076	0.0112	0.0100	0.0101
15	0.0772	0.1039	0.0961	0.0272	1	3.8500e-04	3.8500e-04	0	0.0082	0.0078	0.0082	0.0102	0.0091	0.0093
16	0.1206	0.1033	0.1009	0.0094	0.9630	0.0609	0.5565	0.3830	0.0063	0.0061	0.0065	0.0092	0.0084	0.0079
17	0.1230	0.0998	0.0935	0.0102	1	3.8500e-04	3.8500e-04	0	0.0054	0.0049	0.0056	0.0084	0.0074	0.0078
18	0.0783	0.1026	0.0936	0.0179	0.8085	6.6100e-04	0.0100	0.0054	0.0081	0.0073	0.0080	0.0114	0.0103	0.0107
19	0.1426	0.0993	0.0960	0.0070	1	3.8500e-04	3.8500e-04	0	0.0054	0.0051	0.0057	0.0069	0.0064	0.0080
20	0.1368	0.0982	0.0929	0.0088	1	3.8500e-04	3.8500e-04	0	0.0058	0.0052	0.0053	0.0093	0.0082	0.0094
21	0.0489	0.1028	0.0921	0.0333	1	3.8500e-04	3.8500e-04	0	0.0119	0.0118	0.0113	0.0145	0.0139	0.0135
22	0.0266	0.1027	0.0937	0.0280	0.6182	7.0100e-04	0.0184	0.0109	0.0093	0.0084	0.0096	0.0094	0.0086	0.0091
23	0.1234	0.0988	0.0927	0.0116	0.9608	0.0591	0.5377	0.3674	0.0059	0.0051	0.0059	0.0094	0.0083	0.0085
24	0.1168	0.1008	0.0963	0.0134	0.9453	0.0633	0.5265	0.3789	0.0063	0.0056	0.0064	0.0090	0.0082	0.0092
25	0.1159	0.1018	0.0976	0.0093	0.9427	0.0627	0.5276	0.3801	0.0061	0.0059	0.0066	0.0086	0.0081	0.0098
26	0.1060	0.1019	0.0934	0.0166	0.9558	0.0691	0.5105	0.3641	0.0065	0.0059	0.0067	0.0093	0.0082	0.0087
27	0.0560	0.1019	0.0900	0.0206	0.9518	0.0952	0.4715	0.3547	0.0105	0.0103	0.0102	0.0146	0.0145	0.0136
28	0.1432	0.1041	0.1039	0.0125	0.9398	0.0534	0.5616	0.3730	0.0057	0.0052	0.0056	0.0100	0.0090	0.0093
29	0.1113	0.1045	0.1011	0.0159	0.9602	0.0638	0.5729	0.3888	0.0068	0.0063	0.0065	0.0093	0.0086	0.0080
30	0.1046	0.1039	0.1003	0.0175	0.9721	0.0692	0.5464	0.3821	0.0079	0.0072	0.0077	0.0105	0.0100	0.0095
31	0.0917	0.1033	0.0986	0.0223	0.5556	0.0015	0.0168	0.0099	0.0091	0.0083	0.0085	0.0117	0.0104	0.0104
32	0.0621	0.1005	0.0888	0.0338	0.7120	0.0013	0.0180	0.0117	0.0091	0.0087	0.0091	0.0118	0.0111	0.0109
33	0.0454	0.1098	0.1037	0.0252	0.7643	0.0011	0.0172	0.0100	0.0117	0.0107	0.0114	0.0150	0.0133	0.0138

Figure 7.1: Diagram showing a few sample surviving features from the GA



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
1	0.0579	0.0034	0.0216	0.0044	4.2900e-04	0.9737	0.0549	0.3565	0.1154	0.0051	0.0047	0.0052	0.0086	0.0078
2	0.0563	0.0022	0.0194	0.0042	7.1300e-04	0.9618	0.0383	0.2585	0.1187	0.0029	0.0027	0.0041	0.0078	0.0066
3	0.0590	9.1800e-04	0.0191	0.0030	4.8400e-04	1	3.8500e-04	0	0.1187	0.0024	0.0022	0.0024	0.0086	0.0075
4	0.0594	0.0041	0.0211	0.0099	0.0014	0.9620	0.0656	0.2998	0.1154	0.0048	0.0046	0.0047	0.0075	0.0071
5	0.0541	0.0044	0.0243	0.0177	0.0026	0.9113	0.0853	0.3187	0.1125	0.0062	0.0058	0.0057	0.0090	0.0077
6	0.0606	0.0031	0.0210	0.0023	4.6600e-04	1	3.8500e-04	0	0.1125	0.0071	0.0064	0.0063	0.0111	0.0098
7	0.0612	0.0039	0.0226	0.0023	2.1700e-04	0.9629	0.0714	0.3738	0.1096	0.0074	0.0069	0.0072	0.0106	0.0100
8	0.0616	0.0042	0.0219	0.0034	1.2700e-04	0.9520	0.0638	0.3877	0.1149	0.0068	0.0064	0.0067	0.0086	0.0079
9	0.0549	0.0022	0.0251	0.0084	0.0013	0.9137	0.0758	0.3291	0.1125	0.0070	0.0069	0.0067	0.0099	0.0096
10	0.0570	0.0023	0.0277	0.0082	0.0013	1	3.8500e-04	2.0000e-04	0.1158	0.0076	0.0069	0.0075	0.0103	0.0095
11	0.0623	0.0044	0.0275	0.0053	5.9000e-04	0.9597	0.1022	0.3565	0.1154	0.0104	0.0097	0.0097	0.0131	0.0124
12	0.0627	0.0042	0.0225	0.0039	3.8100e-04	0.9313	0.0765	0.3774	0.1120	0.0076	0.0072	0.0076	0.0087	0.0080
13	0.0554	0.0043	0.0307	0.0106	0.0019	0.6316	7.3400e-04	0.0072	0.1154	0.0104	0.0101	0.0099	0.0127	0.0121
14	0.0609	0.0032	0.0232	0.0014	2.3800e-04	0.9237	0.0733	0.3674	0.1096	0.0076	0.0072	0.0076	0.0112	0.0100
15	0.0636	0.0065	0.0238	0.0070	6.4900e-04	1	3.8500e-04	0	0.1182	0.0082	0.0078	0.0082	0.0102	0.0091
16	0.0608	0.0028	0.0208	8.2400e-04	3.5900e-04	0.9630	0.0609	0.3830	0.1125	0.0063	0.0061	0.0065	0.0092	0.0084
17	0.0595	0.0031	0.0222	8.9500e-04	3.4000e-04	1	3.8500e-04	0	0.1125	0.0054	0.0049	0.0056	0.0084	0.0074
18	0.0627	0.0041	0.0233	0.0027	5.3100e-04	0.8085	6.6100e-04	0.0054	0.1120	0.0081	0.0073	0.0080	0.0114	0.0103
19	0.0583	0.0021	0.0205	0.0030	4.6100e-04	1	3.8500e-04	0	0.1154	0.0054	0.0051	0.0057	0.0069	0.0064
20	0.0580	0.0021	0.0213	0.0017	1.6900e-04	1	3.8500e-04	0	0.1154	0.0058	0.0052	0.0053	0.0093	0.0082
21	0.0642	0.0091	0.0254	0.0059	0.0010	1	3.8500e-04	0	0.1154	0.0119	0.0118	0.0113	0.0145	0.0139
22	0.0648	0.0068	0.0269	0.0078	7.4400e-04	0.6182	7.0100e-04	0.0109	0.1127	0.0093	0.0084	0.0096	0.0094	0.0086
23	0.0589	0.0032	0.0222	7.6600e-04	2.8700e-04	0.9608	0.0591	0.3674	0.1125	0.0059	0.0051	0.0059	0.0094	0.0083
24	0.0597	0.0044	0.0216	0.0027	5.3600e-04	0.9453	0.0633	0.3789	0.1125	0.0063	0.0056	0.0064	0.0090	0.0082
25	0.0603	0.0032	0.0215	0.0015	4.2300e-04	0.9427	0.0627	0.3801	0.1125	0.0061	0.0059	0.0066	0.0086	0.0081
26	0.0614	0.0035	0.0225	0.0025	2.1400e-04	0.9558	0.0691	0.3641	0.1149	0.0065	0.0059	0.0067	0.0093	0.0082
27	0.0634	0.0060	0.0251	0.0022	0.0010	0.9518	0.0952	0.3547	0.1182	0.0105	0.0103	0.0102	0.0146	0.0145
28	0.0604	0.0031	0.0194	0.0025	1.7500e-04	0.9398	0.0534	0.3730	0.1154	0.0057	0.0052	0.0056	0.0100	0.0090
29	0.0619	0.0047	0.0214	0.0042	3.1600e-04	0.9602	0.0638	0.3888	0.1096	0.0068	0.0063	0.0065	0.0093	0.0086
30	0.0617	0.0048	0.0217	0.0037	2.5400e-04	0.9721	0.0692	0.3821	0.1096	0.0079	0.0072	0.0077	0.0105	0.0100
31	0.0621	0.0071	0.0229	0.0035	5.8200e-04	0.5556	0.0015	0.0099	0.1158	0.0091	0.0083	0.0085	0.0117	0.0104
32	0.0631	0.0077	0.0264	0.0089	0.0010	0.7120	0.0013	0.0117	0.1154	0.0091	0.0087	0.0091	0.0118	0.0111
33	0.0671	0.0064	0.0245	0.0049	4.0300e-04	0.7643	0.0011	0.0100	0.1154	0.0117	0.0107	0.0114	0.0150	0.0133

Figure 7.2: Diagram showing a few sample surviving features from the PSO

7.4 Artificial Neural Networks

In this section a brief introduction to Artificial Neural Networks (ANN) is presented. ANNs are abstractions of Human cognitive systems. The main function of human biological neuron (a cell in the brain) is to collect, process, and disseminate electrical signals. The capacity of human brain is traceable to massive networks of these biological neurons (Russel & Norvig, 2003). ANNs are mathematical abstraction or formal specification of these massive neurons (network of neurons) in human brains. In analogy to human brains, ANNs are network of neurons having nodes with connecting links between them. Each node has an associated functional and local parameters such as synaptic weight and bias term. The fundamental basis for designing ANNs is a neuron as shown in Figure 7.3. The basic terms associated with a neuron in all ANNs are given as follows:

1. Input nodes: A neuron receives $R \geq 1$ inputs from the external source via its inputs nodes. The input can be represented as $x = \{x_1, x_2, x_3, ..., x_R\} \in \mathbb{R}^R$. A diagram illustrating a simple neuron taken from (Demuth, Beale, & Hagan, 2013) is shown in Figure 7.3.

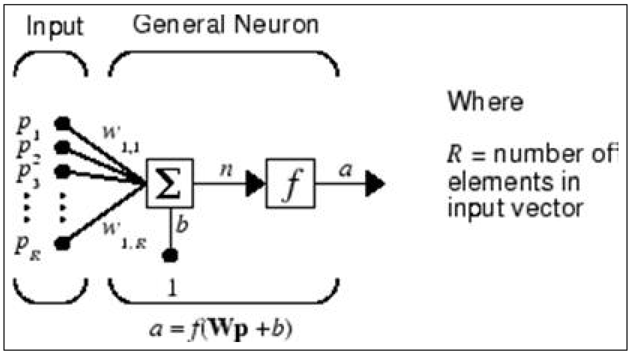


Figure 7.3: A simple Neuronal Model

2. Weights: To every input p_i in a neuron, is associated a weight w_{ij} representing interconnection strength between neuron i and j where i = j = 1(1)k.

3. Inputs Adder: This part of the neuron is responsible for summing all the weights-multiplied input vectors. This can be expressed as $n = \sum_{j=1}^{k} w_{ij} p_j = \mathbb{W}$p.

4. Bias: The neuron has an externally applied bias, say, b which is a special type of weight with constant value of 1. With bias added, the new structure in the adder unit is given as

$n = \mathbb{W}p + b$. A neuron will be active when the condition $f > 0$ holds. The term bias is also called threshold. The value of f is forced to be between certian range such as [0, 1] and [-1, 1] using the neuron activation function. For this thesis, the value of f lies in the closed interval [0, 1] since PNN is probabilistic in nature.

5. Activation function: Activation function (transfer function) of ANN is a functional used in limiting or squashing the amplitude of the output in the ANN (Krose & van der Smagt, 1996; Guo, 2010). In most cases, activation functions are required to be real-valued, continous, and bounded . They can be linear or non-linear. The activation function for neuron shown in Figure 7.3 may be written as $f(n) = f(\mathbb{W}p + b)$. The linearity or non-linearity of the ANN is determined by the transfer function used. Three examples of transfer functions are:

$$f_1(x) = \frac{1}{1+e^{-x}}, 0 \leq f_1(x) \leq 1. \tag{7.4.1}$$

$$f_2(x) = \tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}, -1 \leq f_2(x) \leq 1. \tag{7.4.2}$$

$$f_3(x) = \frac{1}{(2\pi)^{\frac{1}{2}}} \int_{-\infty}^{x} \exp^{\frac{-x^2}{2}} dx \tag{7.4.3}$$

The transfer function in Equation (7.4.2) is often used by many reseachers because it is smooth and well suited for backpropagation algorithm used in training the concerned ANN (Pinkus, 1999). The transfer function in Equation 7.4.2 is a variant of Equation 7.4.1 while the third transfer function (Equation 7.4.3) is used by some ANNs and it's the main transfer function used by PNN to make it probabilistic ANN.

6. Neuron Output: The ouput of the neuron in Figure 7.3 can be written as $a = f(\mathbb{W}p + b) = f(n)$.

7. K. Wang, Chen, and Lau (2011) described a typical feed-forward as a ANN model which has

3 layers viz input, hidden, and output layers with multiple neurons can be represented as

$$y_i = f\left[\sum_{k=1}^{N} \omega_k g\left(\sum_{j=1}^{J} (\omega_j x_j + \phi_j)\right) + \varepsilon_k\right] \tag{7.4.4}$$

where N = Number of hidden-layer neurons, ω_j = synaptic weights connecting the input and hidden layer neurons, ω_k = weights connecting the biases in the hidden and output layers, while $f(.)$ and $g(.)$ are respectively linear and sigmoid functions.

8. Other issues such as learning mode (supervised and unsupervised), network of neurons, feedforwardness and feedbackwardness of the neurons which have been reported for ANN have been described previously in (Bishop, 1995; Russel & Norvig, 2003; Demuth et al., 2013).

7.5 Baye's rule and Bayesian Classifier

Let us assume we are given the dataset as seen in Table 5.1. Each class $c_i, i = 1(1)32$, has a priori probability p_i of occuring in the dataset. The rationality behind the Baye's rule is to compute the a posteriori probabilities from the a priori probabilities and the evidence. The feature space in Table 5.1 can be represented as measurable pair $(X, c_i) \in \mathbb{R}^D \times \{1, 2, 3, ..., 32\}$ where $c_i = 1, 2, 3, ..., 32$ is the class label and D is the number of features in the given input(feature space). This implies that the conditional distribution of X, given that the class information c_i takes on a value $i \in Z^+$ is given by $X|c_i = i \dashrightarrow P_i$ for $i = 1(1)32$ where "$\dashrightarrow$ " means "distributed as ", and where P_i is a probability distribution.

A classifier is a functional that assigns to an observation $X = c_i$, a measure (mostly numeric) of what the class information for the unseen data actually was. In formal (mathematical) notation based on the dataset in Table 5.1, a classifier is a measurable function $ClassMap : \mathbb{R}^D \mapsto \{1, 2, 3, ..., 32\}$, where the D-Dimensional input vectors are mapped to any of the intergers $1, 2, 3, ..., 32$. Bayesian Classifier is a probabilistic classifier model and a probabilistic classifier model (based on Table 7.1) is a conditional model $p(X|c_1, c_2, c_3, ..., c_{32})$ over a dependent class variable c_i, i = 1(1)32, conditioned on several features $(F_1, F_2, F_3, ..., F_n)$, n being the number of features in the dataset.

The definitions (7.5.1 to 7.5.3) are needed to understand Bayesian classifier expressed in definition 7.5.13.

Definition 7.5.1. A Posteriori $\mapsto p(X|c_i)$. This is the probability of X given the evidence c_i. This can also be expressed as

$$posteriori = \frac{prior \times likelihood}{evidence}((Bishop,1995)).$$

Definition 7.5.2. A priori $\mapsto p(X)$. This is the probability of X only.

Definition 7.5.3. Baye's rule is given as :

$$p(c_j|X) = \frac{f(X|c_j)p(c_j))}{f(X)}$$

Definition 7.5.4. A function $f(x)$ is smooth if and only if $f(x) \in C^\infty$, for $x \in \mathbb{R}$.

Definition 7.5.5. Probability Density Functions (PDF or pdf): The PDF is term normally used to denote the probability function for continuous variables. A PDF specifies that the probability of a variable x in the interval $[a,b]$ is given as:

$$P(x \in [a,b]) = \int\limits_a^b p(x)dx. \tag{7.5.1}$$

Definition 7.5.6. Gaussian or Normal Random variable: A random variable X is called normal (or Gaussian) with mean μ and variance σ^2 if for all $-\infty \leq a \leq b \leq \infty$,

$$P(x \in [a,b]) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_a^b \exp^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{7.5.2}$$

Definition 7.5.7. A 1-Dimensional Gaussian PDF is defined as

$$p(x) = f(x|\mu,\sigma) = \frac{1}{\sqrt{(2\pi\sigma)}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), x \in \mathbb{R} \tag{7.5.3}$$

Definition 7.5.8. The expected or average value of variable x under the gaussian pdf is given as

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} N(x|\mu,\sigma^2)x dx = \mu \tag{7.5.4}$$

Definition 7.5.9. A n-Dimensional Gaussian PDF is defined as

$$p(x) = f(x|\mu,\sigma) = \frac{1}{(2\pi)^{n/2}}|S|^{\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^T S^{-1}(x-\mu)\right), x \in \mathbb{R} \tag{7.5.5}$$

where μ = E[x] , i.e the mean vector and σ^2 is the variance.

S = covariance matrix defined as $S = E[(x-\mu)(x-\mu)^T]$, $|S|$ is a matrix determinant (Van der Maaten, Postma, & Van Den Herik, 2009).

Definition 7.5.10. σ-algebra:

A σ-algebra is a collection of U of subsets of Ω (Ω non-empty) with the following properties:

1. $\phi, \Omega \in U$

2. if $A \in U$, then $A^c \in U$

3. if $A_i \in U, i = 1(1)n$, then
   $\bigcup_{k=1}^{\infty} A_k \in U$ and $\bigcap_{k=1}^{\infty} A_k \in U$

Definition 7.5.11. A probability space is defined as a triple $S = (X, U, P)$, where X is a non-empty set, U is a $\sigma - algebra$, and P is a probability measure.

Definition 7.5.12. Let U be a $\sigma - algebra$ of subsets of Ω. Then a map $P : U \to [0,1]$ is called a probability measure provided the following condition holds.

1. $P(\phi) = 0, P(\Omega) = 1$.

2. $P(X_i) \geq 0$ for every $X_1, X_2, X_3, ... \in \Omega$.

3. $P(\bigcup_{i=1}^{\infty} X_i) = \sum_{i=1}^{\infty} P(X_i)$

The Bayesian Classifier is based on finding the class for a given test data X. Once the probability measure for every class c_j in the pattern unit is calculated, the test data X is classified as belonging to the class c_j for which $p(c_j|X)$ is maximized . In order to estimate $p(c_j|X)$, we need to compute the class conditional probabilities $f(X|c_j)$ and the a priori probabilities $p(c_j)$ for every class c_j in the class information. The calculation of $f(X)$ is not compuslory because its value is homogenous across all classes. The a priori probabilities are computed from the training dataset as shown in Algorithm 2. According to Parzen (1962), the class conditional probabilities $f(X|c_j)$ can be computed from the training data using the Equation 7.5.6, which is a variant of Equations in definitions 7.5.7 and 7.5.9.

$$f(X|c_j) = \frac{1}{(2\pi)^{M/2}\sigma^M N_j} \sum_{i=1}^{N_j} \exp\Big(- \frac{(X - X_i^j)^T (X - X_i^j)}{2\sigma^2} \Big) \tag{7.5.6}$$

where

1. M = dimensionality of the features;

2. N_j is Number of training patterns belonging to class j;

3. $X_i^j = i_{th}$ training pattern of X^j;

4. σ is Gaussian smoothing parameter or standard deviation;

The factor $\frac{1}{\sqrt{\pi}}$ in Equation 7.5.6 ensures that the total area under the pdf curve is equal to 1. The $\frac{1}{2}$ in the exponent makes the pdf to have unit variance (and of course, unit standard deviation). The normal curve is symetrical around $x = 0$ where it reaches a maximum value of $\frac{1}{\sqrt{\pi}}$. The points of inflexion for this pdf is at -1 and $+1$.

Definition 7.5.13. The Baye's classifier is thus defined as $CLASSIFIER_{Bayes}(x) = argmax(P(c_i = i|X = x))$

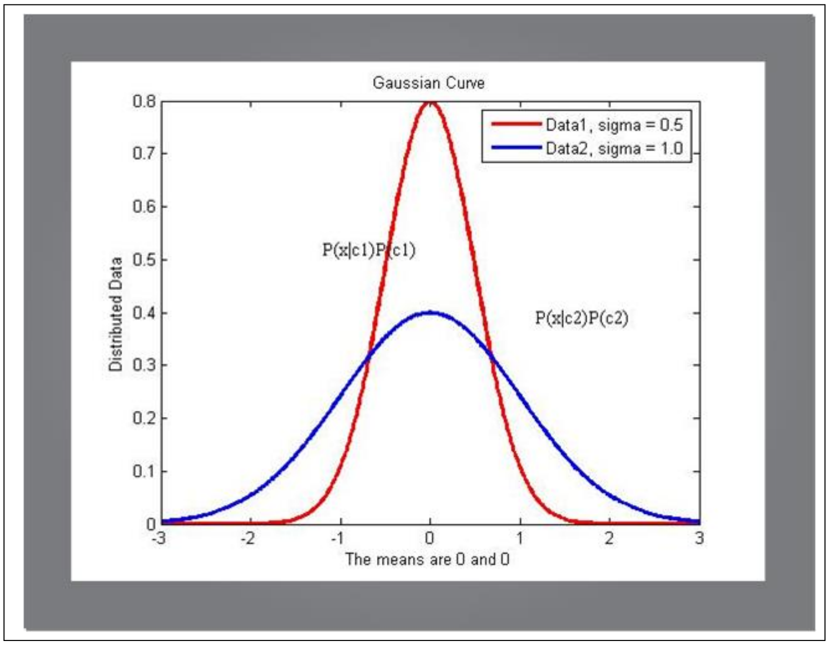


Figure 7.4: Two Gaussian Windows based on different spread (sigmal)

7.6 Probabilistic Neural Networks (PNN)

PNN is a feedforward Neural Network that uses kernel methods for density estimation in a multi-category problem and which was introduced by D. Specht (1967, 1971); D. F. Specht (1988); D. Specht (1990). PNN can be seen as a mathematical interpolation or a parallel implementation of Parzen type classifier model. All Parzen classifiers (non-parametric) use the distances of a test vector with unknown class to all observations in the training set. Then the class with maximum discriminant value is assigned to the test sample. Specht replaced the sigmoid activation function commonly employed in conventional ANN with an exponential function to derive a PNN which can give nonlinear boundaries of decision approximating Bayes optimal classifier (Gish, 1990; Grother, Candela, & Blue, 1997; Cheung & Cannons, 2002; Russel & Norvig, 2003; Gorunescu, 2006; J. S. Wang, Song, & Gao, 2015). In other words, a PNN-based classifier approximates class-conditional probability distributions which can be used for Bayesian decision-making (Definition 7.5.13).

The PNN actually creates a set of multivariate probability densities that are derived from the training set presented to the network. The main advantage of using PNN is its ability to converge to the underlying pdf of the data with only few training samples available. The Equation 7.5.6 is the basis for the PNN. This equation can be written as a Bayes decision function if both training set and test set are normalized and if it is assumed that the number instances in each class are directly proportional to the a priori probability. PNN are architecturrally categorized into a multilayered feed forward network having input layer (representing a predictor variable(feature vector)), hidden layer (normally fed by the input layer), pattern layer(summation layer), and output layer. The number of output nodes here equals to the number of classes in the dataset (see Figure 7.5). The PNN is normally used for classification as follows: Given input vector say, $x_i, i \in \mathbb{Z}^+$, the first layer computes the distance separating the input vector (test dataset) from the training input vectors (training dataset). The outcome of this is a vector whose elements indicate metric proximity (neighborhood norm) of the test data to the training data (Demuth et al., 2013). The second layer the computes the sums of the contribution of each elements of input vector to generate a vector of probabilities $\{p_i\}, i \in \mathbb{Z}^+$ stored in a probability space (7.5.11). The last stage is the application of a transfer function (5) to the result of the second layer followed by the selection the maximum of $\{p_i\}, i \in \mathbb{Z}^+$, producing a value 1 (indicating desired output) for that class and a value 0 (indicating unwanted output) for non-targeted classes.

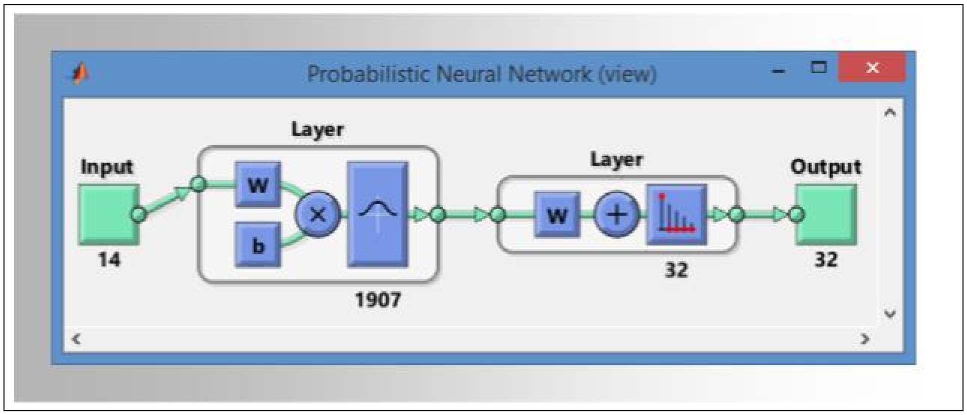


Figure 7.5: A simple view of Probabilistic Neural Networks. The PNN consists of N1 input units, N2 pattern units and N3 category units. The pattern units has the inner product of the weights and training vector and produce the quantity from the activation function. Each unit in the category sums the probabilistic contributions from the pattern unit connected to it

.

7.7 Computational and theoritical properties of PNN

7.7.1 PNN as Neural State Machine

PNN is theoretically defined as a neural state machine or 6-tuple $\chi = (A, B, \sigma, \kappa, \tau, \theta)$ where $A \in \mathbb{R}^{M\times N}$ is the training set containing N features and M observations (with M,N $\in \mathbb{Z}^{+}$), $B \in \mathbb{R}^{m\times N}$ is the test set containing N features and m observations (with $M,N \in \mathbb{Z}^{+}$), κ is the class information such that $\kappa \in R^{c\times 1}$, c = number classes in the dataset, τ is binary vectors such that $\tau \in \mathbb{R}^{c\times 1}$ and has only one "1", $c-1$ "$0s$". The σ is a single real-valued scalar called the PNN spread or Gaussian window and θ is many output real valued functional denoted as $\theta : \{A, B, \sigma, \kappa\} \mapsto \tau$.

7.7.2 Numerical concepts in PNN

The standard training method for PNN requires just a single pass across all the patterns of the training set. Thus PNN is faster than most ANNs since there is no iteration nor computation of weights. The discriminant function for a PNN is Gaussian. Considering a single real-valued variable, the distribution that maximizes the entropy is the Gaussian. In machine learning and statistical computing, Gaussian random variables (hence Gaussian distributions (definitions 7.5.7 and 7.5.9)), are extremely useful for two main reasons :

1. They are good and being used for noise modelling since by central limit theorem (CLT), summations of large independent random variables approaches Gaussian distribution.
2. Secondly, gaussian random variables are easy to manipulate analytically and algebraically (Maleki & Do, 2009)

Definition 7.7.1. A discriminant function for PNN is given as:

$$D_i(y) = \frac{1}{N^{(i)}} \sum_{j=1}^{N^{(i)}} \exp\left(\frac{-1}{2\sigma^2} d(y, x_j^{(i)}) \right) \tag{7.7.1}$$

where

1. d is a distance metric which is commonly Euclidean.

2. $d(y,x) = (y-x)^T(y-x)$ (for L_2 Euclidean squared distance).

3. For L_2 measure (squared distance metric), the kernel function will be normal and through normalization of the discriminant values of each of the classes (by dividing them by their sum), estimates of the a posteriori probabilities are obtained (See definition 7.5.1).

4. $d(y,x) = max(y-x)$ (for L_∞ maximum metrics).

5. $d(y,x)$ can be generalized as $L_p(y,x) = \left(\sum_{i=1}^{d} |x_i - y_i| \right)^{1/p}$ and it's also called L_p norm.

6. y is a test data. (e.g a given image of unknown class species to be classified. (see Figure 7.8))

7. $x_j^{(i)}$ is a reference training pattern.

8. $N^{(i)}$ = number of training patterns belonging to class i (see Table 7.3).

9. σ is a smoothing parameter which is the Gaussian standard deviation and thus, determines the performance of the PNN. This is evidenced in Figure 7.11, where the accuracy of the PNN varies with the smoothing parameter or the spread. Niculescu, Lewis, and Tigner (2008) refers to PNN spread as "parameter for sphere of influence".

10. An unknown y is mapped to class which has maximum discriminant value $D_i(y)$. The two important properties of discriminant value $D_i(y)$ are given as expressions 11 and 12. These two properties determine the values of the probilities involved in the application of PNN.

11. $$\lim_{d(y,x_j^{(i)}) \to +\infty} D_i(y) = 0.$$

12. $$\lim_{d(y,x_j^{(i)})\to 0} D_i(y) = +\infty.$$

13. Decision Regions: These are subset $R_1, R_2, R_3, ...$ representing partitioning of the entire training set in Table 7.3 using the class information .

14. The input vector classified is based on the shortest distance between inputs and distribution functions (training set) specific to a class. The decision regions form a partition of the feature space F as illutrated in Equations 7.7.2 & 7.7.3.

$$R_i \cap R_j \neq \phi, \ \ i \neq j \tag{7.7.2}$$

$$\cup R_i = F \tag{7.7.3}$$

Misclassification occurs whenever there are similar sets in the decision regions.

7.7.2.1 Neuronal model in PNN

Definition 7.7.2. A PNN neuron with k inputs mapping a normed linear space $X_{INPUT} \subset \mathbb{R}^k$ of input signals (a k-neuron on X_{INPUT}) is a function :

$$F : \mathbb{R}^k \times X_{INPUT} \ni (w,x) \mapsto F(w,x) = ActFUNC(< w,x >) \in \mathbb{R} \tag{7.7.4}$$

where w is a weight vector associated with the training set, $< .,. >$ is a real scalar product, and $ActFUNC : \mathbb{R} \mapsto \mathbb{R}$ is called the activation function of the neuron.

7.7.3 Using PNN for Image (or Pattern) Classification

Let $\{X_i, Y_i\}_{i=1}^k$, $X_i \in \mathbb{R}^k, Y_i \in c_j, j = 1(1)N_c$, N_c = number of classes in the dataset and k = number of observations in the training set and $X_i \in A \subset \mathbb{R}^k$, be sequence of features generated from the

Flavia dataset. The PNN classifier in this context is defined as:

$$PNN_{CLASSIFIER} : \mathbb{R}^k \times \{\mathbb{R}^k \times \{1,2,3,...,N_c\}\}^n \mapsto \{1,2,3,...,N_c\} \tag{7.7.5}$$

where k is the number of feature set (k = 14 for this GA-based features), n = number of observations in the dataset (e.g n = 1907 for this study).

The problem is to estimate Y from X and γ_i, where $\gamma_i = \{X_i, Y_i\}_{i=1}^k$ is a learning sequence with associated activation function shown in mapping 7.7.4. Suppose that p_n and $f_n, n = 1(1)M$, are the priori class probabilities and the class conditional densities, respectively (see Equations 7.5.1 & 7.5.2), we define a discriminant function as shown in Equation 7.7.1 also as follows:

$$D_j(x) = p_j f_j(x) \tag{7.7.6}$$

Let $H(i,j)$ be the loss incurred in taking action $i \in c_j$ when the class is j. The assumption here is $0-1$ loss function since PNN employs ingredients from probability measure (see definition 7.5.12).

Given a decision function $\Phi : \mathbb{R}^k \mapsto c_j$, the performance of Φ can be measured by the conditional probability of error expressed as

$$H(\Phi_n) = p\{\Phi_n(X : \{X_i, Y_i\}^n{}_{i=1}) \neq Y : \{X_i, Y_i\}^n{}_{i=1})\} \tag{7.7.7}$$

while the associated loss is:

$$R(\Phi) = \sum_{j=1}^{M} p_j \int_A H(\Phi(x), j) f_j) dx \tag{7.7.8}$$

A decision function Φ^* which classifies every $x \in A$ as coming from any class n for which :

$$p_n f_n = max\{p_j f_j(x)\} = max\{D_j(x)\} \tag{7.7.9}$$

is a Bayes-decision function (see Definition 7.5.13) and

$$R^* = R(\Phi^*) = \sum_{j=1}^{M} p_j \int_A H((\Phi^*), j) f_j(x) dx \tag{7.7.10}$$

is the minimal Bayes risk. The function $D_j(x)$ is called the Bayes-discriminant function.

Suppose n_j is the number of observations from class $j, (j = 1(1)M)$, the observations in the training set are partitioned into the following subsequences:

$$X_1{}^1, ..., X_{n_1}{}^1$$

$$X_1{}^2, ..., X_{n_2}{}^2$$

$$....$$

$$X_1{}^M, ..., X_{n_M}{}^M \tag{7.7.11}$$

An estimates of conditional denisties f_j can then be given in the form:

$$\hat{f}_{n_j}(x) = \frac{1}{n_j} \sum_{i=1}^{n_j} K_{n_j}(x, X_i{}^j). \tag{7.7.12}$$

The K in Equation 7.7.12 is a guassian kernel of the form shown in definition 7.5.3. The prior probabilities p_j are estimated as

$$\hat{p_j} = \frac{n_j}{n} \tag{7.7.13}$$

The algebraic combination of Equations 7.7.6, 7.7.12, & 7.7.13 gives the following discriminant function estimate:

$$\hat{D}_{j,n}(x) = \frac{1}{n} \sum_{i=1}^{n_j} K_{nj}(x, X_i^{(j)})(7.7.14)$$

and the associated classification procedure

$$\hat{\Phi}_n(x) = m \ \ if \ \ \sum_{i=1}^{n_m} K_{nm}(x, X^{(m)}{}_i) \geq \sum_{i=1}^{n_j} K_{nj}(x, X^{(j)}{}_i) \ \ for \ \ i \neq m, i = 1, 2, 3, ..., M \tag{7.7.15}$$

7.7.4 PNN Optimal Decision Theory

Definition 7.7.3. Borel Measure:

Let X be a locally compact Haurdorff space, and let $\mathbb{B}(X)$ be the smallest σ-algebra that contains the open set of X; this is known as the σ-algebra of Borel sets. Any measure μ imposed on the σ-algebra of Borel sets is called a Borel measure.

For a measurable set $A \in \mathbb{R}^k$, the probability measure μ is given as:

$$\mu(A) = p\{X \in A\} \tag{7.7.16}$$

and for any $x \in \mathbb{R}^k$,

$$\rho(x) = p\{Y = i, i = 1(1)N_c | X = x\} = E\{Y|X = x\} \tag{7.7.17}$$

ρ is the conditional probability that Y is $i, i = 1(1)N_c$ given $X = x$.

For any $Q \subseteq \mathbb{R}^k \times \{1,2,3,...,N_c\}$, we have

$$Q = \bigcup_{i=1}^{N_c}(Q \bigcap(\mathbb{R}^k \times \{i\})) \equiv \bigcup_{i=1}^{N_c}(Q_i \times i) \tag{7.7.18}$$

and

$$p\{(X,Y) \in Q\} = \sum_{i=1}^{N_c} p\{X \in Q_i, Y = i\} = \sum \int_{Q_i} (1 - \rho(x)\mu(dx) \tag{7.7.19}$$

Theorem 7.7.4. For any decision function

$\Phi_n : \mathbb{R}^k \mapsto \{1,2,3,...,N_c\}$,

$p\{\Phi^*(X) \neq Y \leq p\{\Phi(X) \neq Y\}$

7.8 MATLAB Implementation of PNN

In this section a general description of the PNN from MATLAB Neural Network Toolbox and its application to Plant species classification are presented. The general description on PNN Toolbox is given in section 7.8.1 while the application of the PNN Toolbox is given in section 7.8.2.

7.8.1 Description of PNN Classifier

The architecture of the PNN is shown in Figure 7.5. It has input layer, pattern layer (radial basis layer), competitive Layer and output Layer.

1. Input Layer: The number of nodes in the input layer depends on the cardinality of the input vectors (or predictor variable). Each input vector has an associated classes in the class information given in the dataset.

2. Pattern Layer: Each of the pattern (entries in the traning set) is normalized to have a unit norm. i.e for all the descriptors or features $x_i, i = 1(1)N$, $N=$ the number of features in the training set, the condition $\sum_{i=1}^{N} = 1$ must be met. One of the merits of PNN is the training speed since their is no training for the weights as $w_i = x_i$ is quite simple and only requires a single pass through the training set. Once the input are fed into the PNN via the input layer, the pattern layer computes the metric norm between the input vector and each vectors in the training set. This norm is based on the discriminant function in Equation 7.7.1. At this layer, the norms for each class in the training pattern are added together to form a set of probability measures (probability space). If a given input vector produces similar metric norms for several training vectors, it is then represented by several entries in the probability space of the PNN. These entries all close to 1. The cardinality of this probability space is the same as the number of classes in the training set.

3. Competitive Layer: With the given probability measures in pattern layer, the competitive layer applies a transfer function on each probabilities and select the maximum of these probabilities.

4. Output Layer: The PNN classifies the given input vector (test data) into a specific class that has the maximum probability. This is the similarity between PNN and Bayesian Classifier. At the output layer, a 1 is produced for the particular class with maximum probability and a 0 for the other class. For example, the output vector {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} implies there are 32 classes in the given dataset and object belonging to the class indexed by the position of 1 is classified. In the

given example, the species belonging to class 4 is predicted since the positional index of 1 in the binary strings is 4.

The MATLAB PNN Toolbox can used for classification by following the Pseudocode given in Algorithm 6.

Algorithm 6 Application of PNN for Classification

```
procedure PNNCLASSIFY
    Input: Input DataSet TrainingSet and the class information ClassInfo.
    Input: Extract the number of patterns in each classes and store them in a variable, say, T.
    Set the value of PNN spread (smoothing parameter).
    Set a variable, say, net = PNNCLASSIFIER(TrainingSet, ClassInfo, Spread).
    Get a TestImage TestImage.
    Extract the same number of features in DataSet from TestImage and store them in TestImageFeatures.
    Output = simulate (net, TestImageFeatures)
end procedure
```

7.8.2 Plant Species Classification Using PNN

The Neuro Genetic Hybrid Intelligent System (NGHIS) shown in Figure 7.8 or 7.9 was built using MATLAB and PNN Toolbox. The PNN will normally approach the Bayes classifier if enough training samples are provided. The Gaussian windows with different spread values for the PNN were similar to that shown in Figure 7.4. The window with wider spread value of 1.0 has wider decision space and will be prone to more classification error than that with narrower spread value. The value of spread is normally assumed to be small or around $\frac{1}{n(c)}$, where $n(c)$ is the number of classes in the dataset (Wu et al., 2007; Bao, Lie, & Zhang, 2008; Han, Embrechts, & Szymanski, 2011). The transfer function for the PNN is a smooth function (See 7.5.4). The smoothness herein helps the PNN to be continous and also guarantees convergence and unique solution for any given input. The number of nodes in the pattern layer is equal to the number of training instances. The number of nodes in the summation layer is the same as the number of classes in the training set. The input layer is followed by the pattern layer. The input layer only accepts feature vectors and supplies the input vectors to the neurons in the pattern layer. The pattern layer is transitively connected to the summation layer. The summation units only sum the input vectors from the pattern units that correspond to the class information from which the training pattern was selected. The cardinality of non-colour features selected by the GA is 11 while that selected from the second PhD dataset (non-colour + colour features) was 14. The input node of the PNN is seen to be of size 14 in Figure 7.5 while the second layer has 1907 showing the number of observations in the original Flavia dataset input into the PNN in Figure 7.5. This is the exact number of images in the Flavia Dataset which we used for this work. If the loaded dataset is the training set used, then the number 1907 in the figure will be 1587. It is at first layer that distances of all observations in the training set from test set are computed. The second layer shows a numerical value of 32 indicating the number of classes (and decision regions (13)) in the DataSet. The PNN smoothing parameter was set to be 0.025 for a start and this drastically improved the classification accuracy. The smoothing parameter will further be optimized in Chapter 8 of this thesis. The dimensions of both Training dataset and Test dataset was 1587×14 and 320×14 respectively. More precisely, the Training set was generated from the 1587 images of the Flavia Dataset whose details is show in Table 7.3 while the Test Dataset was generated from another 320 set of images which are disjoint from the training set. The detailed operations of the PNN is also shown in Algorithm 7.

Algorithm 7 Pseudocode for PNN Classifier

```
procedure PNNClassify(TrainingSet, TestSet, Spread)
    Input TrainingSet and the class information.
    Normalize the vectors in the dataset to have unit norm. This applies to both training samples and test samples as well. The normalization ensures that $\sum_{i=1}^{d} {x_i}^2 = 1$ for all features $x_i$.
    Compute $N_i, p(c_i)$, where $N_i$ = Number of training patterns in each class $c_i$ = class information, i = 1(1)32.
    Input PNN Spread as $\sigma \mapsto$ 0.025 and set counter = 1
    Pick an Observation $X_{test}$ from TestSet
    DO counter ↦ counter + 1
    In the pattern unit, compute unconditional probability p($X_{test}$) and conditional probability p($X_{test}|c_i$) respectively as
        $p(X_{test}) = \frac{1}{(2\pi)^{M/2}\sigma^M M} \sum_{i=1}^{M} \exp\left[-\frac{(X_{i,k}-X_{test})^T(X_{i,k}-X_{test})}{2\sigma^2}\right]$   (7.8.1)
        $p(X_{test}|c_i) = \frac{1}{(2\pi)^{M/2}\sigma^M N_i} \sum_{i=1}^{N_i} \exp\left[-\frac{(X_{i,k}-X_{test})^T(X_{i,k}-X_{test})}{2\sigma^2}\right]$   (7.8.2)
    where $M$ = total number of observations in the training set, $i$ is the (vector) pattern number, $X_{test}$ = test dataset, $X_{i,k}$= $k_{th}$ training vector from plant species of class $c_j$ with $j = 1(1)32$, $k = 1(1)N_i$, $\sigma$ = PNN spread or smoothing parameter
    Compute posteriori probability of $X_{test}$ as
        $p(c_i|X_{test}) = \frac{p(X_{test}|c_i)p(c_i)}{p(X_{test})}$
    Compute the average of inputs from pattern units as
        $f_i = \frac{1}{N} \sum_{i=1}^{N} p(c_i|X_{test})$   (7.8.3)
    where $N_i$ = Number of training patterns belonging to class $c_i$.
    UNTIL counter = M
    The classification of each pattern vector is made according to the Baye's Rule:
        $i = argmax\ \{f_i\}$   (7.8.4)
end procedure
```

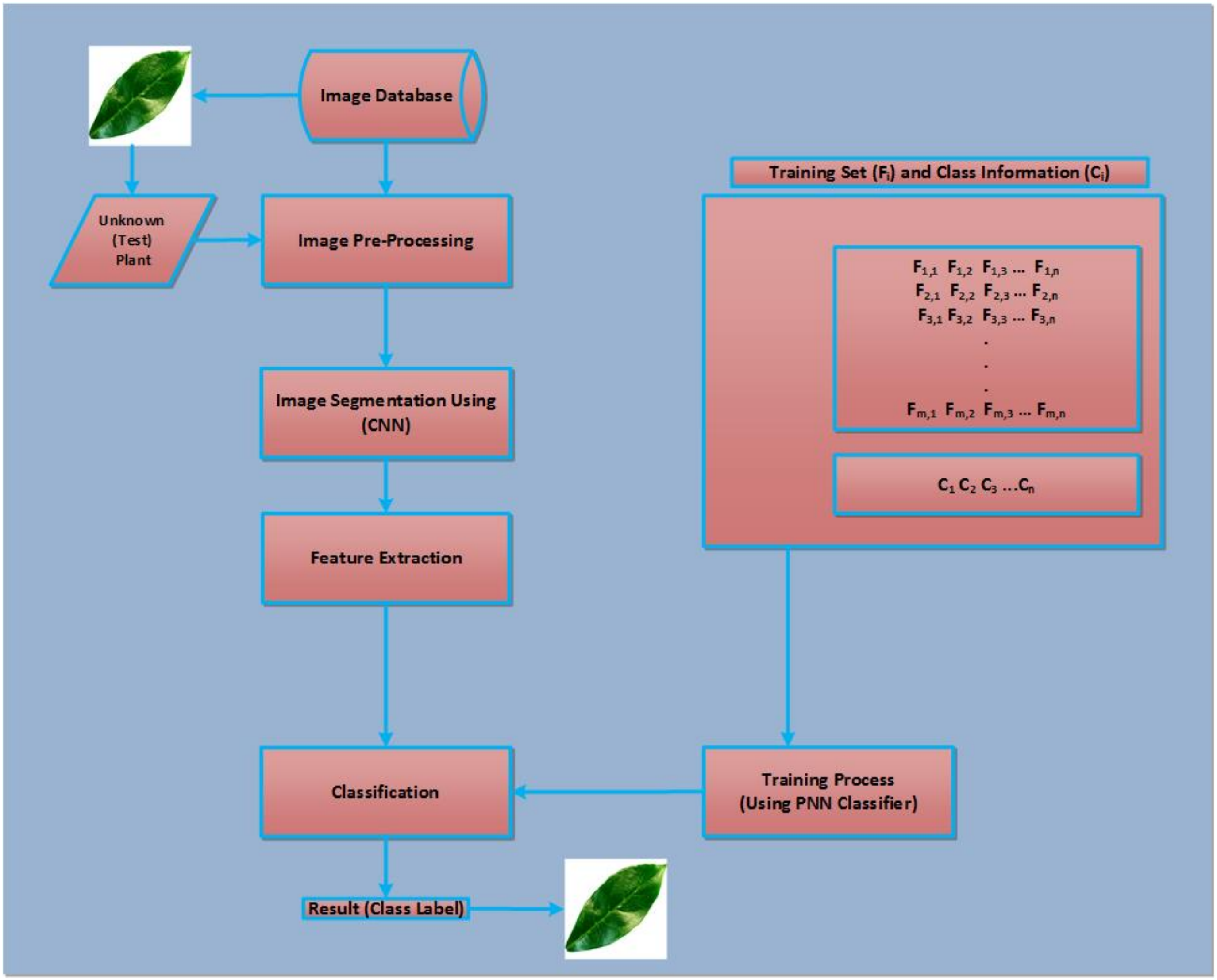
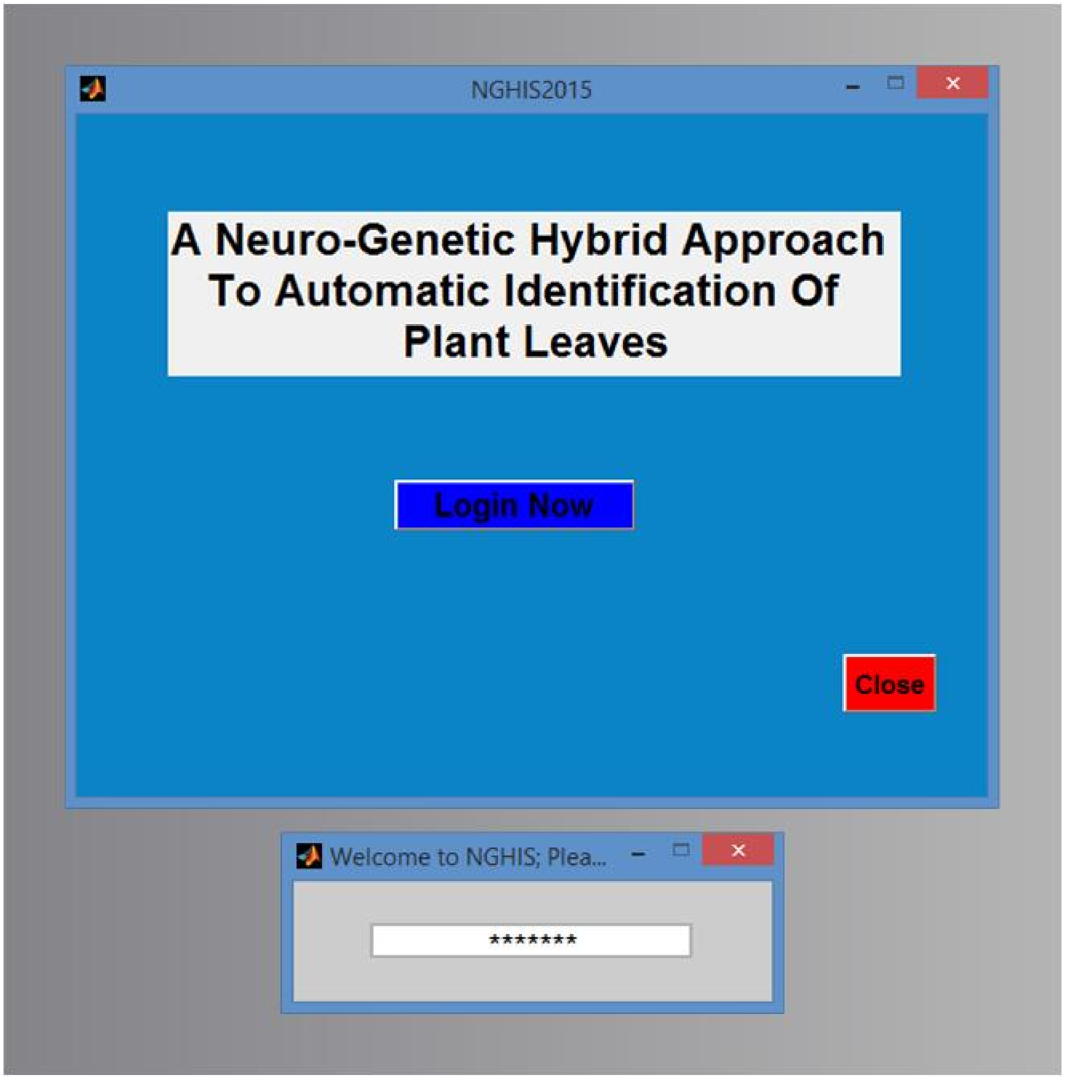


Image Database
Unknown (Test) Plant
Image Pre-Processing
Image Segmentation Using (CNN)
Feature Extraction
Classification
Result (Class Label)
Training Set (F_i) and Class Information (C_i)
$F_{1,1}$ $F_{1,2}$ $F_{1,3}$ ... $F_{1,n}$
$F_{2,1}$ $F_{2,2}$ $F_{2,3}$ ... $F_{2,n}$
$F_{3,1}$ $F_{3,2}$ $F_{3,3}$ ... $F_{3,n}$
$F_{m,1}$ $F_{m,2}$ $F_{m,3}$ ... $F_{m,n}$
C_1 C_2 C_3 ...C_n
Training Process (Using PNN Classifier)

Figure 7.6: Learning system based on PNN Classifier

7.9 Implementation of PNN-based Image Classification System

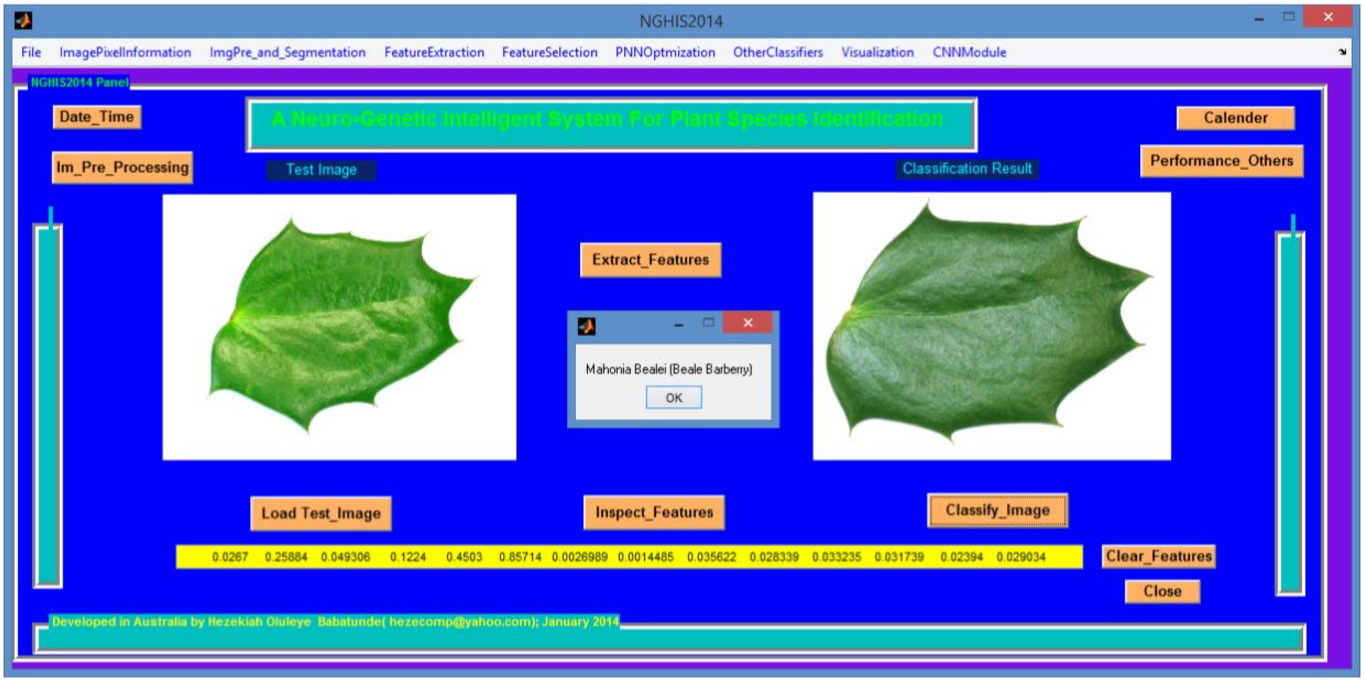
The implementation of the image classification system is shown in Figure 7.6. The logic in this Figure is similar to others used in this study. The PNN spread was arbtrarily set to be equal to 0.025. This value was motivated by the fact the PNN spread should be chosen to in the neighborhood of $R_c = \frac{1}{N_c}$ where N_c is the number of class information in the training set. The actual value of R_c is $\frac{1}{32} = 0.03125$. As seen in Figure 7.6, the design of the PNN-based classification system shown in Figure ?? comprises of image acquisition (from the Flavia dataset), extraction descriptors from the images to form the training set. Several techniques needed before the formation of the training set were image preprocessing (resizing, colour-to-gray conversion, gray to binary conversion), extraction of Zernike Moments (20 ZMs), Fourier Descriptors (21 FDs), Legendre Moments (20 LMs), Hu 7 Moments (7 Hu7Ms), Texture Properties (22 TP) , Geometrical Properties (10 GP), and Colour features (12 CF). The image features (descriptors) were extracted in a batch mode processing and stored as a 1907 × 112 matrix of real numbers. To classify an unknown species, the cardinality (same number) of features found in the training set is should be extracted from the test set to avoid dimensionality conflict and for the PNN classifier to work.



A Neuro-Genetic Hybrid Approach To Automatic Identification Of Plant Leaves

****

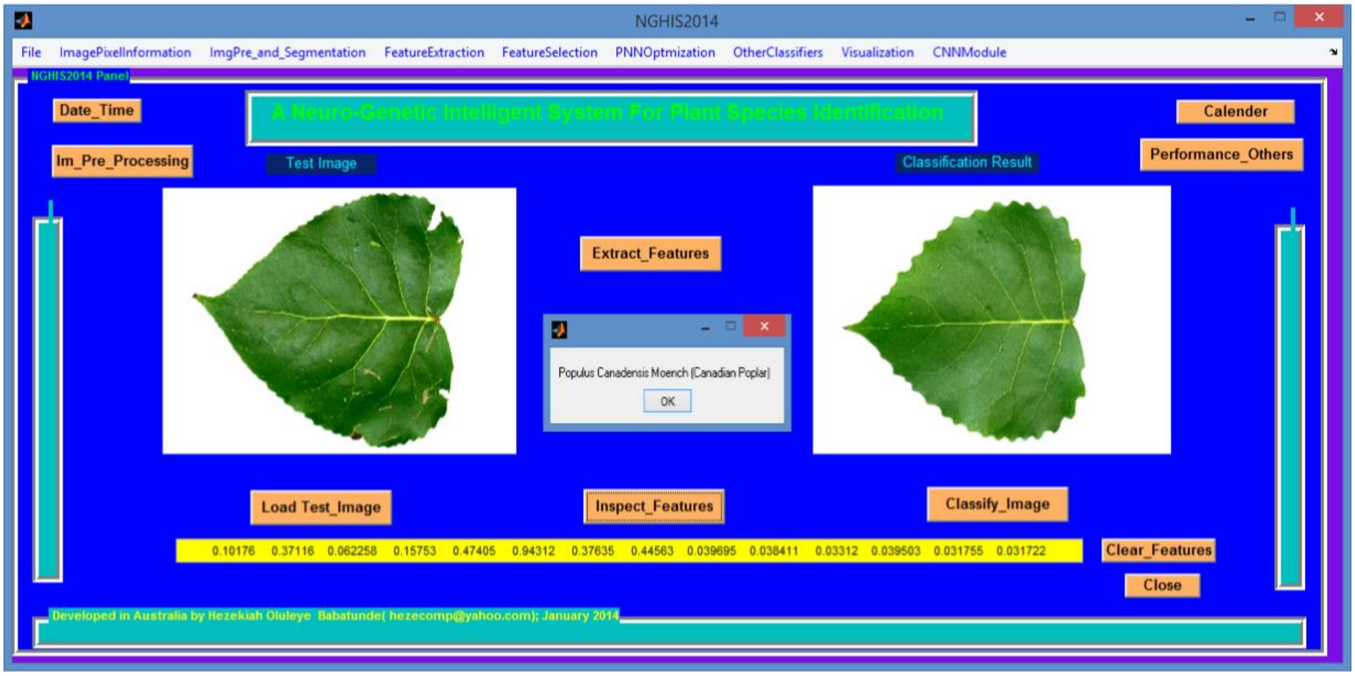
Figure 7.7: Login interface to the classification system developed



Mahonia Bealei (Beale Barberry)

```
0.0267   0.25884   0.049306   0.1224   0.4503   0.85714  0.0026989  0.0014485   0.035622   0.028339   0.033235   0.031739   0.02394   0.029034
```

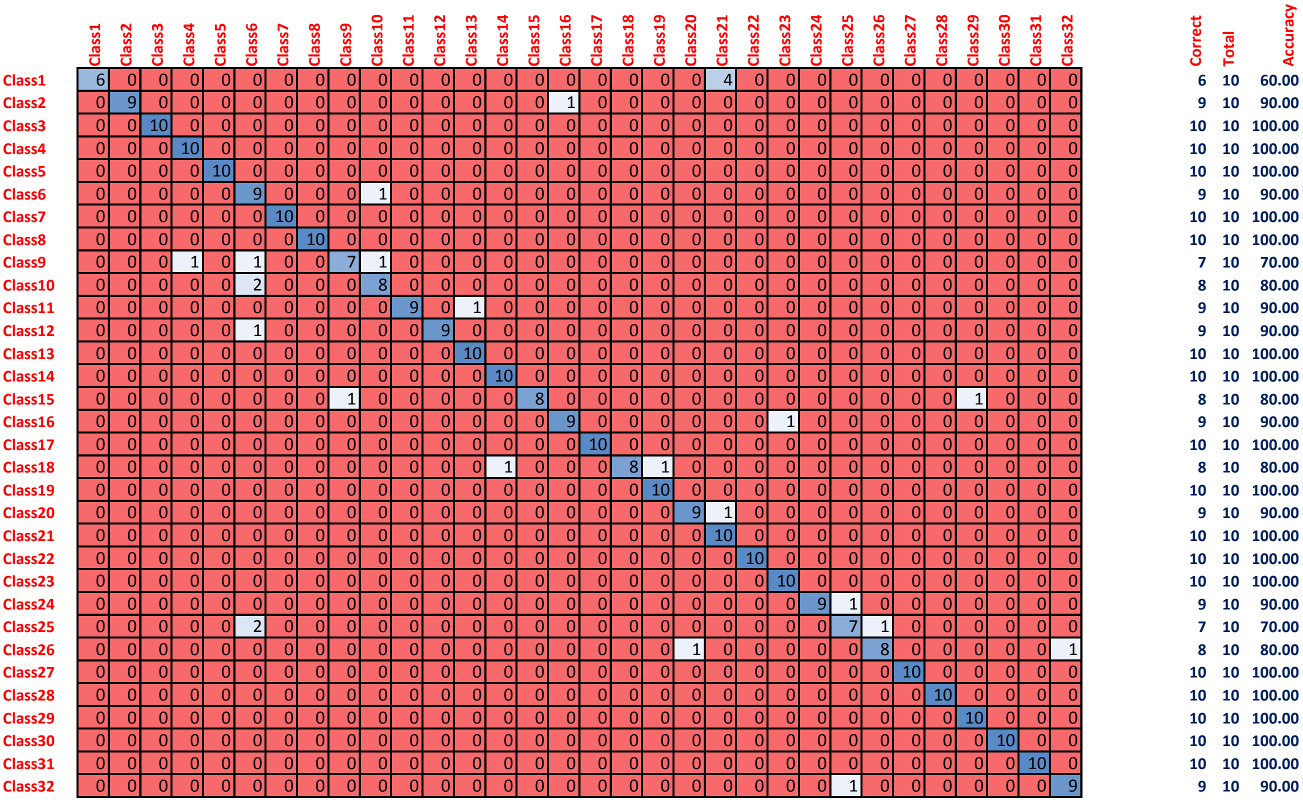
Figure 7.8: Classification Result using an unknown Plant Species as Test Image



Populus Canadensis Moench (Canadian Poplar)

```
0.10176   0.37116   0.062258   0.15753   0.47405   0.94312   0.37635   0.44563   0.039695   0.038411   0.03312   0.039503   0.031755   0.031722
```

Figure 7.9: Another Classification Result using an unknown Plant Species as Test Image



	Class1	Class2	Class3	Class4	Class5	Class6	Class7	Class8	Class9	Class10	Class11	Class12	Class13	Class14	Class15	Class16	Class17	Class18	Class19	Class20	Class21	Class22	Class23	Class24	Class25	Class26	Class27	Class28	Class29	Class30	Class31	Class32	Correct	Total	Accuracy
Class1	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	6	10	60.00
Class2	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	10	90.00
Class3	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class4	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class5	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class6	0	0	0	0	0	9	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	10	90.00
Class7	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class8	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class9	0	0	0	1	0	1	0	0	7	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	10	70.00
Class10	0	0	0	0	0	2	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	10	80.00
Class11	0	0	0	0	0	0	0	0	0	0	9	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	10	90.00
Class12	0	0	0	0	0	1	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	10	90.00
Class13	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class14	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	8	10	80.00
Class16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	9	10	90.00
Class17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	8	1	0	0	0	0	0	0	0	0	0	0	0	0	0	8	10	80.00
Class19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	1	0	0	0	0	0	0	0	0	0	0	0	9	10	90.00
Class21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	10	10	100.00
Class23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	10	10	100.00
Class24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	1	0	0	0	0	0	0	0	9	10	90.00
Class25	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	1	0	0	0	0	0	0	7	10	70.00
Class26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	8	0	0	0	0	0	1	8	10	80.00
Class27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	10	10	100.00
Class28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	10	10	100.00
Class29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	10	10	100.00
Class30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	10	10	100.00
Class31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	10	10	100.00
Class32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	9	9	10	90.00

Figure 7.10: Confusion Matrix for PNN Classifier based on Flavia Dataset

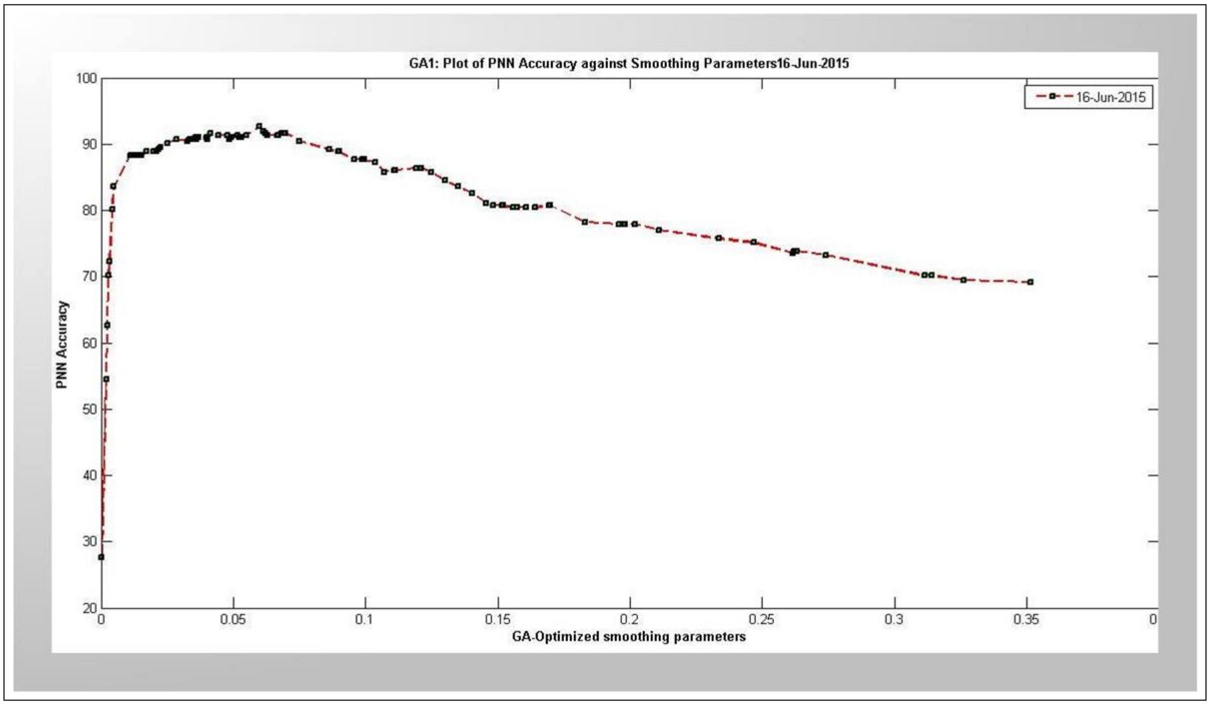


Figure 7.11: PNN Accuracy Versus PNN Spread

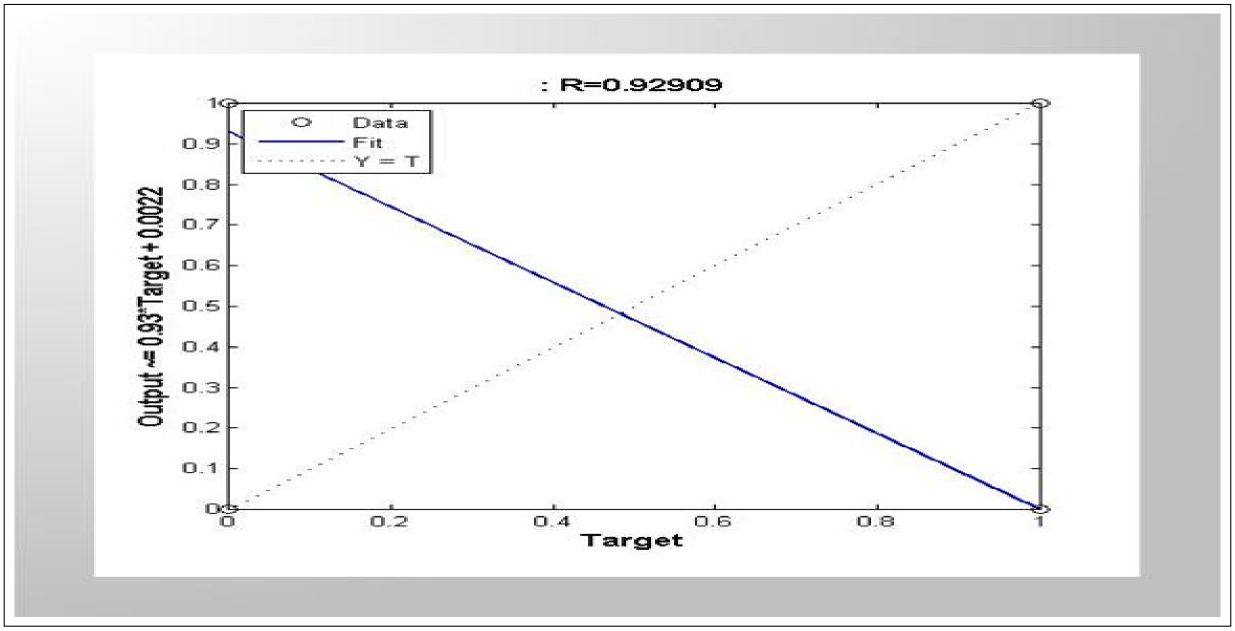


Figure 7.12: Regression of Predicted Species on the Actual Species

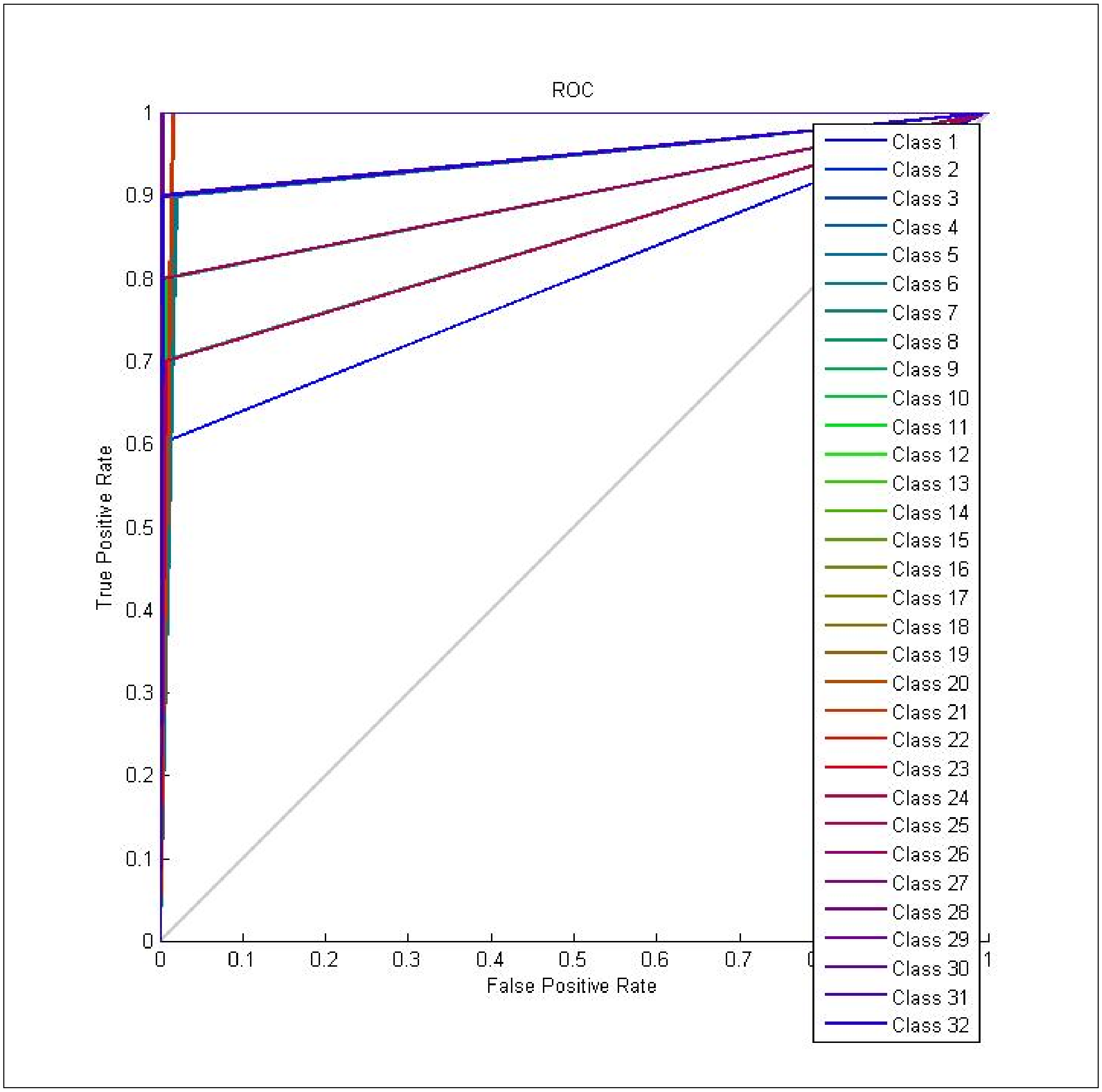


Figure 7.13: Receiver Operating Characteristics Curve for 32 Classes

7.10 Experimental Validation

To validate the correctness of the PNN classifier, four(4) approaches were used to test the correctness of the system. It is to be noted that there is no training or computation of weights in PNN. Training PNN is done just by placing the TrainingSet in the PNN input layer. The only parameter that are tuned by researchers in this field is the PNN spread (smoothing parameter) which is the standard deviation for the Gaussian distribution involved in using the PNN. This can be set to a fixed value or optimized by some optimization technique during the validation process. For this section, a fixed value 0.025 is chosen for the PNN spread as shown in Algorithm 7. The reason for using this fixed value, as discussed earlier in Section 7.8.2, is to make sure its value is epsilon-neighborhood of $\frac{1}{n(c)}$, where $n(c)$ is the number of classes in the dataset (Wu et al., 2007; Bao et al., 2008; Han et al., 2011). In the next chapter, the value of PNN parametr will be optimized using GA. The transfer function for the PNN is a smooth function (See 7.5.4). The approaches used in validating our NGHIS are listed and explained below:

(1) Experimental testing using a single unknown plant species.
(2) 10-Fold CrossValidation.
(3) Confusion Matrix computation.
(4) Regression Analysis.

1. First Experiment : Experimental testing using a single unknown plant species as shown in Figures 7.8 & 7.9

   The first approach involved using the image of an unknown specie and testing it against the training set TrainingSet. The TrainingSet was generated from the Flavia Dataset shown in Section 7.2 and Table 5.1. Herein, the same type and number features in the training set was extracted from the image of the unknown species and fed into the PNN classifier. This approach was used in the GUI shown in Figures 7.8 & 7.9 with methodology appearing in Figure 7.6. A user of the system shown in Figure 7.8 needs to load the image of an unknown plant species via the Load Image button. Once the image is loaded into the system, the user then clicks the ExtractFeature button and finally, the Classify Image button. For the purpose of validation, screen shots of two images selected are shown in Figures 7.8 & 7.9, displaying correct classification of the species shown. The steps involved in creating Experiment 1 is given as follows:

STEP 1: Input TrainingSet

STEP 2: Set PNN Smoothing Parameter (spread)

STEP 3 Pass TrainingSet and spread as arguments to the PNN function from MATLAB Toolbox.

STEP 4: Get image ImgUnseen of unknown (unseen) species to be classified.

STEP 5: Extract TestImageFeatures from ImgUnseen.

STEP 6: Simulate PNN with TestImageFeatures.

STEP 7: Display Classification Result.

2. Second Experiment : k-Fold Cross validation (kFCV)

   The second approach used in validating the PNN Classifier is the k-Fold cross validation (k-Fold CV) with k = 10. Generally, a cross validation (CV) is a method of partitioning the feature space into training and testing sets (see Figure 7.15). CV is also called rotation estimation. In this study, the models were fitted using training set, while the fitted models were validated through testing (or validation) set by measuring the error predicted. The training set and testing set were both disjoint to ensure that the testing set for evaluating the model (in our case, PNN Classifier), were not used in fitting the model (Clarke, Fokoue, & Zhang, 2009). Let the feature space associated with the Table 5.1 be represented as a measurable pair $(X, c_i) \in \mathbb{R}^D \times \{1,2,3,...,32\}$ where $c_i = 1,2,3,...,32$ is the class label and D is the number of features in the feature space. The dataset (feature space)X was then partitioned into two sets viz $X = X_1 \cup X_2$, such that k elements are in X_1 and $D-k$ elements in X_2 (See Figure 7.15). The PNN classifier was then trained or fitted using the set X_2.

The historical pattern of X_2 was then used to produce classifications (predictions) results for observations $X_{X_1} \in X_1$ given X_2. The possible number of partitions we can get from the feature space is given as:

$$\binom{D}{k} = \frac{D(D-1)...(D-k+1)}{k(k-1)(k-2)...1} = \frac{D!}{k!(D-k)!} \tag{7.10.1}$$

The feature space (dataset) X was partitioned into k subsets that are roughly of the same size. This partitioning may be written as $X = \bigcup_{i=1}^{k} X_i$ where each of the subset is called a fold. Thus, there were k folds derived from partitioning the original set (feature space) X. The PNN was trained on $k-1$ folds while the k_{th} fold was used for testing. The procedure is repeated such that each subset (fold) was used only once for testing (See Algorithm 8). The generally recommended value for k is 5 or 10. The k choice for this study was 10. The fascinating merit of the k-Fold CV was that all the observations in the original dataset (feature space) were eventually used for both training and testing. The CV method is much more accurate determinant of the classifier. Using a stratified 10-fold CV, the feature space (newly generated from the original Flavia dataset), was partitioned into training data and test data as {1717 1716 1716 1716 1716 1716 1716 1716 1717 1717} and {190 191 191 191 191 191 191 191 190 190} respectively, and with the PNN Classifier, an average accuracy of in 91.06% was reported. The steps involved in creating Experiment 2 as shown Algorithm 8, is given as follows:

STEP 1: Input DataSet

STEP 2: Partition DataSet into TrainingSet and TestSet using 10-Fold CV

STEP 3: Set PNN Smoothing Parameter (spread = 0.025)

STEP 4 Train PNN with the TrainingSet.

STEP 5: For each fold, simulate PNN with the associated TestSet and Computate CV Error.

STEP 6: Add all the CV Errors from the 10 folds and compute their average

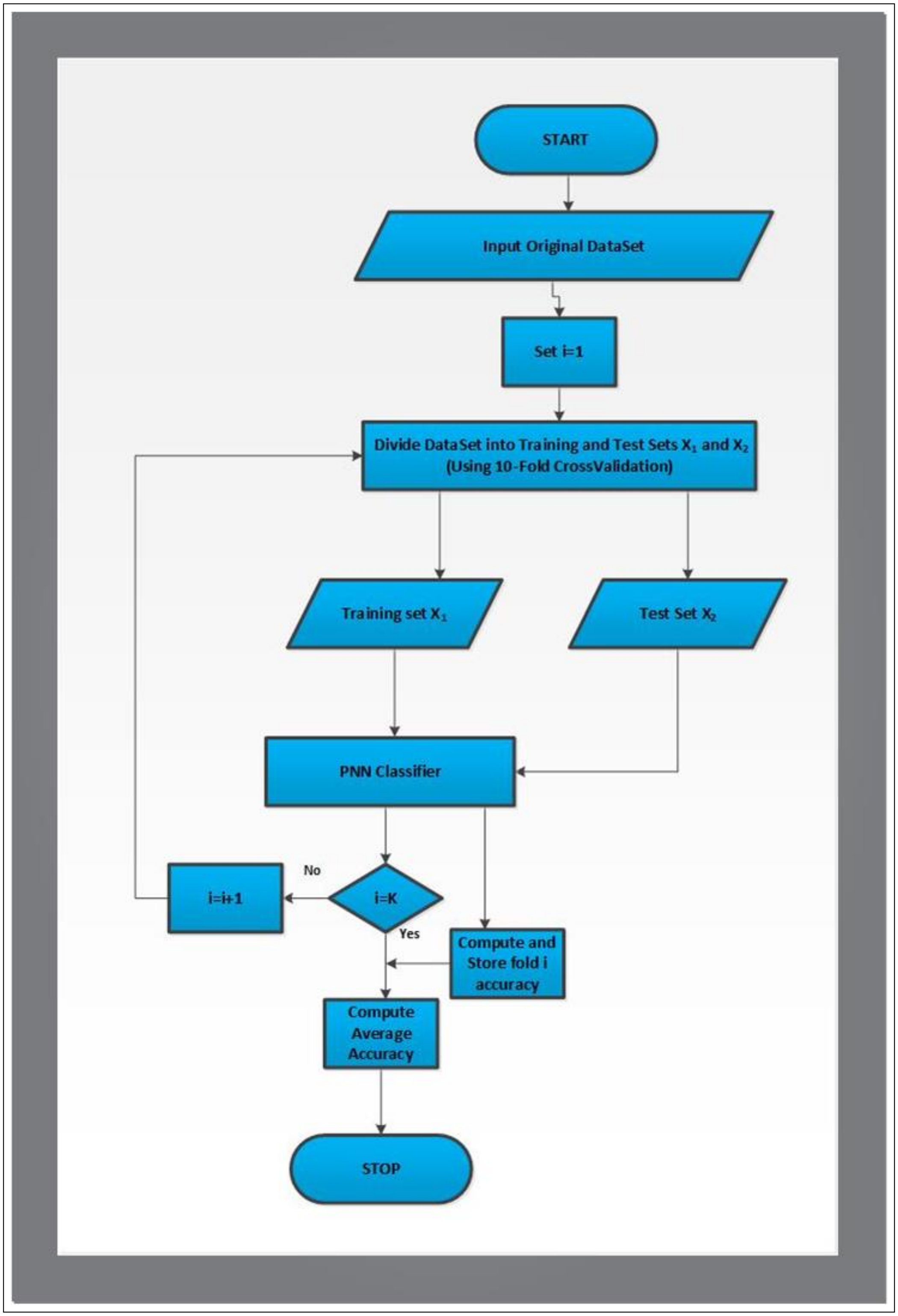

START
Input Original DataSet
Set i=1
Divide DataSet into Training and Test Sets X1 and X2 (Using 10-Fold CrossValidation)
Training set X1
Test Set X2
PNN Classifier
No
i=i+1
i=K
Yes
Compute and Store fold i accuracy
Compute Average Accuracy
STOP

Figure 7.14: Experimental Validation Using K-Fold CV

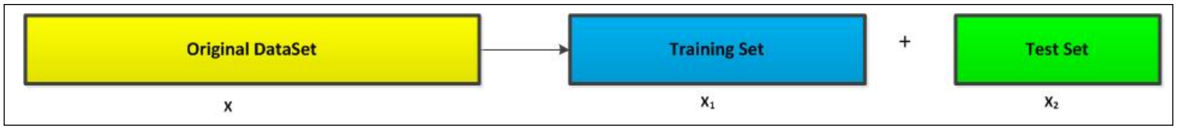


Figure 7.15: Splitting of Original DataSet into Training Set and Test Set

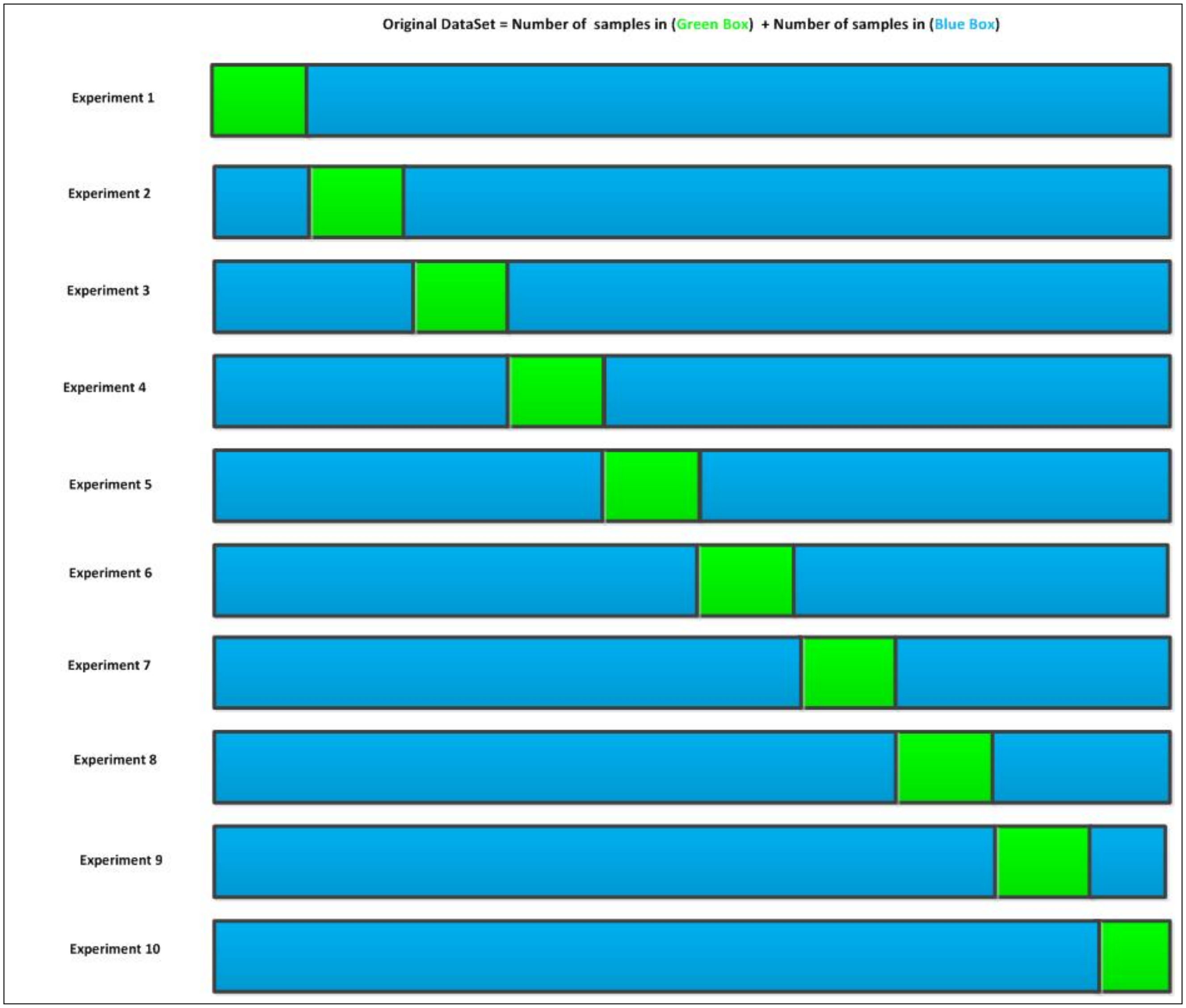


Figure 7.16: Visual Representation of 10-Fold Cross Validation Experiments. The 10-Fold CV runs for 10 iteration, computing the classification accuracy for each fold , storing the accuracies and finally computing the average of these accuracies.

Algorithm 8 k-fold Cross Validation for PNN Classifier

```
procedure kFoldPNN(DataSet, k)
    Input DataSet and the class information.
    Randomly partition DataSet into K folds (disjoint sets) using the class information such as $X = X_1 + X_2$, where $(X, c_i) \in \mathbb{R}^D \times \{1,2,3,...,32\}$
    DO counter $\mapsto$ counter + 1
    Remove k and train PNNClassify using feature from all classes except class k
    Use $X_2$ for validation and $X_1$ for Training
    Compute $Error_{PNN}$ on the validation set $X_2$ as
        $Error_{PNN}(X) = PNNClassify(X_2)$
    j = number of datapoints in the partition k
    UNTIL counter = K
        $CV_{Error} = \frac{1}{N}\sum_{j=1}^{N} Error_{PNN}$
end procedure
```

3. Third Experiment: Confusion Matrix Computation.

   In this section, the performance of the clasifier PNN is analysed using confusion matrix (7.10). A confusion matrix is a tabular tool or matrix display of the instances from the training set that were correctly and incorrectly predicted by the (PNN) classifier. It can be represented as ConfuseMatrix $\in \mathbb{R}^{c \times c}$, a square matrix whose (backward) diagonal elements depicts the actual classification accuracy and c is the number of classes in the dataset (See Figure 7.10). A confusion matrix is also called contigency table or error matrix since it's all about visualising the performance of the learning algorithm. Based on Table 7.3, an entry in ConfuseMatrix is the number of observations of plant species or class c_i that the classifier (PNN) predicts to be of class c_j, where i = j = 1(1)32. Table 7.3 and Figure 7.17 show the performance metric for the PNN classifier using confusion matrix. The training set is used to model the probabilistic classifier. The definitions in Table 7.2 are referenced from Umbaugh (2011) and MathWorks (2013). The classification results based on the confusion matrix are shown in Table 7.3 and Figures 7.10, 7.17. Classification accuracy shown in Figure 7.10 were computed using the equations 7.10.2 & 7.10.3. The average classification accuracy reported by the PNN classifier based on the confusion matrix for the entire dataset was 91.87%. The steps involved in creating Experiment 3 are given as follows:

STEP 1: Input DataSet

STEP 2: Separate DataSet into TrainingSet and TestSet

STEP 3: Set PNN Smoothing Parameter (spread)

STEP 4 Pass TrainingSet and spread as arguments to the PNN function from MATLAB Toolbox.

STEP 5: Simulate PNN with TestSet.

STEP 6: Compute ConfuseMatrix (See 3)

Definition 7.10.1. The accuracy of the PNN classifier (or any other classifier) is defined as

$$PNN_{accuracy} = \frac{n(correct_{classification})}{n(total_{observation})} \tag{7.10.2}$$

Definition 7.10.2. The accuracy of the PNN classifier (or any other classifier) can also be defined as

$$PNN_{accuracy} = \frac{trace(ConfuseMatrix)}{sum(ConfuseMatrix)} \tag{7.10.3}$$

where $trace(.)$ is the sum of all the elements in the backward diagonal, and $sum(.)$ is the sum of all the entries in ConfuseMatrix.

Definition 7.10.3. Receiver Operating Characteristics (ROC): For any classifier, the ROC is the graphical plot of True Positive Rate (TPR) against False Positive Rate (FPR) or sensitivity against (1-specificity). TPR is the same thing as sensitivity and FPR + specivicity = 1. The ROCs for the classifier used in this thesis is shown in Figure 7.13. In the ROCs figure shown, TPR of all classes is ploted against the FPR of all classes. The varying parameters along each ROC is TPR and FPR of all the number of pattern (instances) in each class. As can be seen in Table 7.3, the number of observations (instances) for each species of plants varies. The TPR of all ROC curves generated by the PNN for the 32 classes all lie between 0.67 and 1. The average TPR for all the classes is 0.9301 while average FPR for all the classes is 0.0699. A perfect classifier should have {(0, 1)} for this ordered pair. The same pair for this work has value {(0.0699, 0.9301)}. This indicates a good performance for the classification model in this study.

Table 7.2: Definition of terms used for performance metric (Babatunde et al, 2014)

S/N	Terms	Definition
1	TP	This is the number of actual positive instances that were correctly classified by the classifier as positive
2	TN	This is the number of actual negative instances that were correctly classified by the classifier as negative.
3	FP	This is the number of actual negative instances that were incorrectly classified by the classifier as positive.
4	FN	This is the number of actual negative instances that were .
5	Sensitivity	For any classifier, the sensitivity is defined as the ratio $\frac{TP}{TP+FN}$. The sensitivity indicates the success rate for a particular wanted class or all the species in the class that the PNN correctly classified.
6	Specificity	For any classifier, the specificity is defined as the ratio $\frac{TN}{TN+FP}$. The specificity indicates a numeric measure for those object (in this case plant species) not in the class wanted
7	Recall	For any classifier, recall rate is defined as $\frac{TP}{TP+FN}$.

8	Precision	For any classifier, recall rate is defined as $\frac{TP}{TP+FP}$. Thus, precision can be defined as the ratio of the true values for a class to all the samples found to be in that class by the classifier.
9	Perfect Classifier	A perfect classifier is that which is 100% sensitive and 100% specific.

4. Fourth Experiment : Regression analysis: Finally, the regression plot (linear regression of the target with respect to the output of the classifier) for the 32 species is shown in Figure 7.12. A good classifier should have the R coefficient in the regression curve close to value 1. The equation in the regression plot given as $y = 0.93x + 0.0022$, where y = output, x = target, indicates that a good accuracy since the value of R is 0.93 (very close to 1). The steps involved in creating Experiment 3 is given as follows:

   STEP 1: Input DataSet

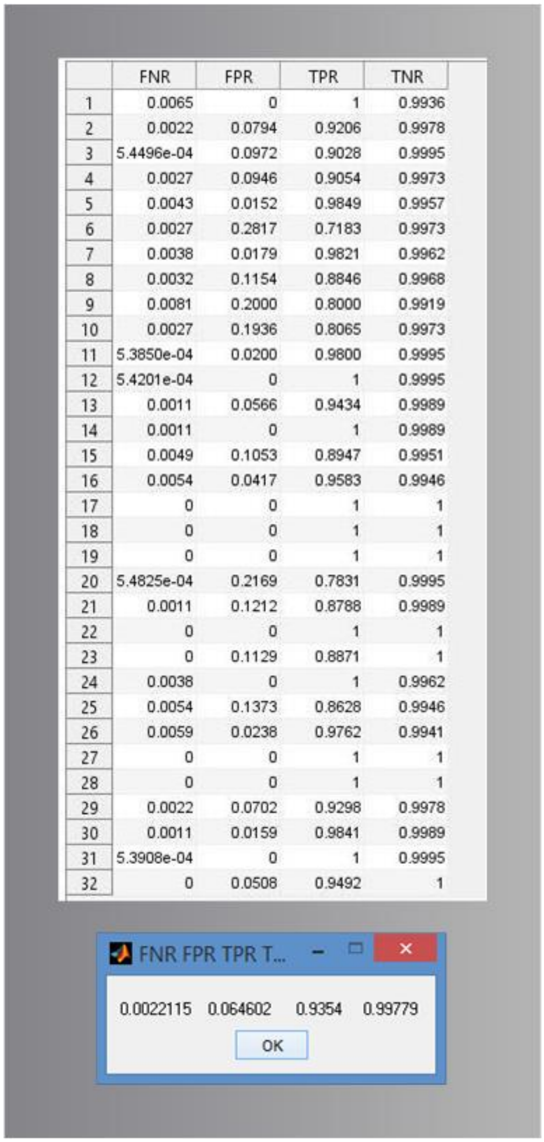
   STEP 2: Separate DataSet into TrainingSet and TestSet

   STEP 3: Set PNN Smoothing Parameter (spread)

   STEP 4 Pass TrainingSet and spread as arguments to the PNN.

   STEP 5: Simulate PNN with TestSet $\mapsto$ {Targets, Outputs} .

   STEP 6: Plot RegressionCurve (Targets, Outputs) (See 7.13)



	FNR	FPR	TPR	TNR
1	0.0065	0	1	0.9936
2	0.0022	0.0794	0.9206	0.9978
3	5.4496e-04	0.0972	0.9028	0.9995
4	0.0027	0.0946	0.9054	0.9973
5	0.0043	0.0152	0.9849	0.9957
6	0.0027	0.2817	0.7183	0.9973
7	0.0038	0.0179	0.9821	0.9962
8	0.0032	0.1154	0.8846	0.9968
9	0.0081	0.2000	0.8000	0.9919
10	0.0027	0.1936	0.8065	0.9973
11	5.3850e-04	0.0200	0.9800	0.9995
12	5.4201e-04	0	1	0.9995
13	0.0011	0.0566	0.9434	0.9989
14	0.0011	0	1	0.9989
15	0.0049	0.1053	0.8947	0.9951
16	0.0054	0.0417	0.9583	0.9946
17	0	0	1	1
18	0	0	1	1
19	0	0	1	1
20	5.4825e-04	0.2169	0.7831	0.9995
21	0.0011	0.1212	0.8788	0.9989
22	0	0	1	1
23	0	0.1129	0.8871	1
24	0.0038	0	1	0.9962
25	0.0054	0.1373	0.8628	0.9946
26	0.0059	0.0238	0.9762	0.9941
27	0	0	1	1
28	0	0	1	1
29	0.0022	0.0702	0.9298	0.9978
30	0.0011	0.0159	0.9841	0.9989
31	5.3908e-04	0	1	0.9995
32	0	0.0508	0.9492	1

```
FNR FPR TPR T...

0.0022115   0.064602   0.9354   0.99779
```

Figure 7.17: Performance Metrics Based on Confusion Matrix. FNR = False Negative Rate = 0.0022, FPR=False Positive Rate = 0.0646, TPR=True Positive Rate = 0.9354, TNR=True Negative Rate = 0.9978

Table 7.3: Comparing our results with Wu's results based on the same Flavia dataset. The numbers of misclassification for both works are shown in Columns 4 and 6 respectively

Class	Botanical Name	PhD	PhD	Wu	Wu
Class1	Phyllostachys Edulis	49	4	58	0
Class2	Aesculus Chinensis	53	1	63	0
Class3	Berberis Anhweiensis	55	0	58	0
Class4	Cercis Chinensis	62	0	72	1
Class5	Indigofera Tinctoria	63	0	72	0
Class6	Acer Palmatum	46	1	53	1
Class7	Phoebe Nanmu	52	0	60	1
Class8	Kalopanax Septemlobus	42	0	51	0
Class9	Cinnamomum Japonicum	49	3	51	2
Class10	Koelreuteria Paniculata	45	2	57	0
Class11	Ilex Macrocarpa Oliv	40	1	50	0
Class12	Pittosporum Tobira	53	1	61	1
Class13	Chimonanthus Praecox	42	0	51	2
Class14	Cinnamomum Camphora	55	0	61	3
Class15	Viburnum Awabuki	50	2	58	2
Class16	Osmanthus Fragrans	46	1	55	5
Class17	Cedrus Deodara	67	0	65	3
Class18	Ginkgo Biloba	52	2	57	0
Class19	Lagerstroemia Indica	51	0	57	0
Class20	Nerium Oleander	56	1	61	0
Class21	Podocarpus Macrophyllus	50	0	60	0
Class22	Prunus Serrulata	45	0	50	0
Class23	Ligustrum Lucidum	45	0	52	0

Class24	Tonna Sinensis	55	1	58	2
Class25	Prunus Persicapeach	44	3	50	2
Class26	Manglietia Fordiana Oliv	42	2	50	3
Class27	Acer Buergerianum Miq	43	0	50	1
Class28	Mahonia Bealei	45	0	50	0
Class29	Magnolia Grandiflora	47	0	50	0
Class30	Populus Canadensis	54	0	58	3
Class31	Liriodendron Chinense	43	0	50	0
Class32	Citrus Reticulata	46	1	51	0

7.11 Results and Discussion

The results of the experiments are shown in Table 7.3, Figures 7.8, 7.9, 7.12,7.10, & 7.17. In comparism to the work by Wu et al. (2007), our system with more images in the dataset and more discriminative features achieved a better average classification of 91.06%. The number of samples for each species are not the same for both work but there are similarities in the results obtained. Wu et al. (2007) conducted their experiments on 1800 images and achieved average classification accuracy of 90.31% while our's was conducted on 1907 images using more discriminative features as shown in Table 7.3. The training set consists 14 features from 1587 images while the test set consists of 14 features from 320 images. There 10 samples per each species in the test set. The main distinguishing factor between our work and that of Wu et al. (2007) is the number and types of features used by the PNN classifier. It is the only reason accountable for the differences between this study and Wu et al. (2007).

The parameters of PNN is further optimized in the next chapter to improve the accuracy of the entire system. Table 7.3 shows classification details and results from this PhD work and the work by Wu et al. (2007). The number of species perfectly classified (100% accuracy) for this work is 16 while that of Wu et al. (2007) is 17. This system in this study was built to be more robust and with improved classification accuracy. Some of the species in Table 7.3 are very similar in shape and as a result of this, some of them were wrongly classified as belonging to another species. Among such are those of Class 16 (Osmanthus Fragrans), Class 19 (Cinnamomium Japonicum), and Class 26 (Mangletia Fordiana Olive). Figure 3.10 from chapter 3 shows the images of the whole leaves in the standard flavia set (which are used as test set in this work) .The shapes of leaves belonging to Class 8, Class 10, and Class 4 are similar. The PNN classifier wrongly classified 4 instances belonging to Class 1 as 4 instances of class 21. The species in Class 9 were wrongly classified as species in Classes 4, 6 and 10. Other classes that were wrongly classified as belonging to another species are fully shown in Table 7.3. The effect of the PNN spread (smoothing parameter) in Figure 7.4) on the performance of the PNN classifier is also demonstrated in Figure 7.11, where the classification accuracy of the PNN varies with the values of the spread. The classification accuracy approaches 100% as the value of the spread moves close to zero. A spread value of 0 is not prefarable to avoid overfitting and to show the true classification ability of the PNN. Also, the PNN becomes approximately equal to kNN (k Nearest Neighbor) in functionality when the spread is taken as zero. As part of the contribution, genetic algorithm was used to obtain combinatorial set of features which were just 14 out of the 112 features computed originally. The GA-selected features improved the performance of the PNN classifier used in this work. To validate the NGHIS (Neuro Genetic Hybrid Intelligent Ssytem), using a 10-fold CV, the feature space was partitioned into training data and test data resulting in 83.29% classification accuracy. Further metrics used for the PNN here are ROC, regression curve, recall (sensitivity) and precision. The ROC for the classifier is shown in Figure 7.13 with most values for True Positive Rate (TPR) lying between 0.67 and 1.00. This is an indicator that our classification system is good. Geometrically, most of the ROCs for the species of plant used lie in the upper left corner of Figure 7.13. The upper left corner is the coodinate (0,1) and sometimes called perfect classification point.

The regression plot (linear regression of the target with respect to the output of the classifier) for the 32 species is shown in Figure 7.12. A good classifier should have the R coefficient in the regression curve close to value 1. The equation in the regression plot is given as $y = 0.93x + 0.0022$, where y = output, x = target, indicates that a good accuracy since the value of R is 0.92 (very close to 1).

The confusion value (the fraction of species misclassified) for the PNN classifier is 0.0813. This implies 26 out of 320 instances were wrongly classified. As shown in Figure 7.17, the average values for False Negative Rate (FNR), False Positive Rate, True Positive Rate (TPR), True Negative Rate (TNR) for all the classified species of plants are respectively given as { 0.0022,0.0646, 0.9354, 0.9978}. Again these values indicate the classification strength of this system.

Chapter 8

Optimization of PNN Smoothing Parameter Using Genetic Algorithm

8.1 Introduction

This chapter details the effect of Gaussian smoothing parameter (spread) on the performance of Probabilistic Neural Networks (PNN). It is considered to be problematic to fix this parameter arbitrarily. Figure 8.1 shows four different slider-based PNN spread values as well as their associated classification accuracies. The slider values 0 & 1 show the two lowest classification accuracies. The optimum accuracy will thus be between 0 and 1. Using the slider to get the maximum accuracy is tiresome and stressful. There is need to automatically determine the optimal accuracy.

In this chapter, two (2) different Genetic Algorithms (GAs) were used to optimize the PNN spread in order to avoid under and over fitting so as to detect the PNN spread that gives the maximum accuracy based on the given training set. The functionalities of the PNN depend on the standard deviation of the underlying Gaussian dustribution. This parameter, commonly called PNN spread or smoothing parameter is a determinant of the receptive width of the Gaussian window for the pdf of the training set. In other words, the value of the PNN spread is fully dependent on the available training sets. This implies the PNN spread value varies across specific

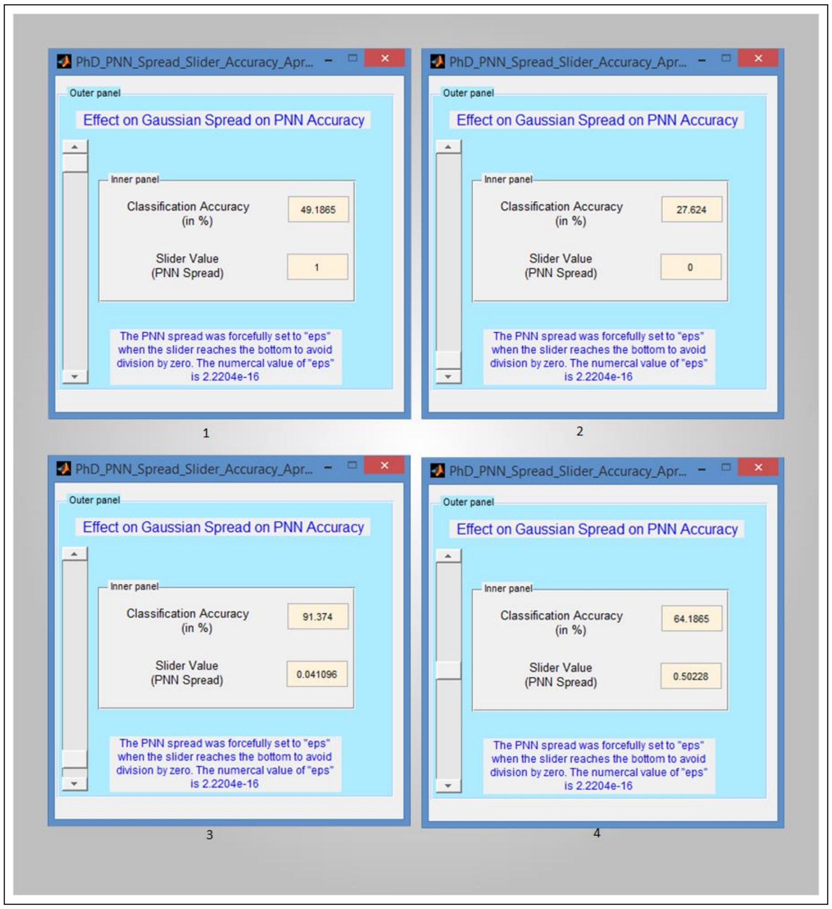
Table 8.1: Parameters Used in MATLAB GA Toolbox

GA Parameter	Value
Population size	100
Genomelength	1
Population type	real
Fitness Function	PNN-Based Classification Error
Number of generations	100
Number of GA Iteration	1
Crossover	Heuristic Crossover
Crossover Fraction	0.8
Mutation	Uniform Mutation
Mutation Fraction	0.01
Selection scheme	Tournament of size 2
EliteCount	2

application since the dataset may not be the same among these dataset.

A common approach to fetch parameters from probability distribution using an observed data set is to find the parameter values that maximize the likelihood function. Similarly, GA was used to minimize the classification error of the PNN by globally searching for the spread value that gives the minimum classification error. The training set was divided into two disjoint sets, each of which contains 1587 images and 320 images for both training and test set respectively. The spread, when it is too small, can cause the PNN to overfit (be very selective), since each training data point will have too much influence and when it is too large, can cause the PNN to be under selective. Striking the balance between underfitting and overfitting is the main rational for tuning the PNN spread.

In this section, the focus is on how to use GA (see Algorithm 9) to optimize the smoothing parameter (spread) of the PNN Classifier to further improve the classification accuracy of the PNN. The parameters setting for the GA are shown in Tables 8.1 & 8.2. The GA developed in this study improved the performance of the PNN. This chapter (see Figure 8.5) also serves as a framework for building image classification or pattern recognition system.


PhD_PNN_Spread_Slider_Accuracy_Apr...

Outer panel

Effect on Gaussian Spread on PNN Accuracy

Inner panel

Classification Accuracy (in %) 49.1865

Slider Value (PNN Spread) 1

The PNN spread was forcefully set to "eps" when the slider reaches the bottom to avoid division by zero. The numercal value of "eps" is 2.2204e-16

1

PhD_PNN_Spread_Slider_Accuracy_Apr...

Outer panel

Effect on Gaussian Spread on PNN Accuracy

Inner panel

Classification Accuracy (in %) 27.624

Slider Value (PNN Spread) 0

The PNN spread was forcefully set to "eps" when the slider reaches the bottom to avoid division by zero. The numercal value of "eps" is 2.2204e-16

2

Outer panel

Effect on Gaussian Spread on PNN Accuracy

Inner panel

Classification Accuracy (in %) 91.374

Slider Value (PNN Spread) 0.041096

The PNN spread was forcefully set to "eps" when the slider reaches the bottom to avoid division by zero. The numercal value of "eps" is 2.2204e-16

3

Outer panel

Effect on Gaussian Spread on PNN Accuracy

Inner panel

Classification Accuracy (in %) 64.1865

Slider Value (PNN Spread) 0.50228

The PNN spread was forcefully set to "eps" when the slider reaches the bottom to avoid division by zero. The numercal value of "eps" is 2.2204e-16

4

Figure 8.1: Variation of PNN accuracies with slider-based smoothing parameters

Table 8.2: Parameters Used in our GA

GA Parameter	Value
Population size	100
Genomelength	1
Population type	real
Fitness Function	PNN-Based Classification Error
Number of generations	100
Number of GA Iteration	1
Crossover	Arithmetic Crossover
Crossover Probability	0.8
Mutation	Gaussian Mutation
Mutation Probability	0.01
Selection scheme	Tournament of size 2
EliteCount	2

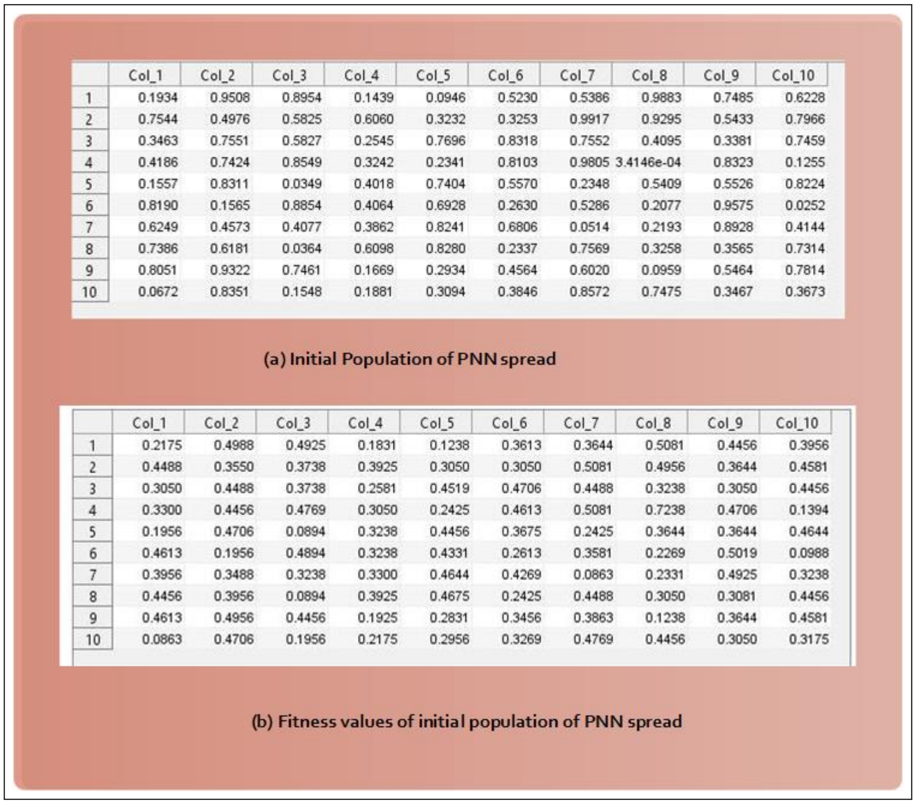
1. Chromosome encoding: Direct decimal encoding was employed to represent the smoothing parameter. To this end, a set of real random numbers are generated in the GA to represent the initial population. The dimension of the initial population is PopSize × GenomeLength, where GenomeLength = 1, PopSize = 100 for both Table 8.1 & 8.2 configuration

Algorithm 9 Generation of initial population

```
procedure GenerateInitialPOP()
    Repeat
EvaluateFitnessFunction ()
SelectionFunction ()
GeneticReproduction ()
    Until Termination condition is satisfied.
    Return Optimal solutions.
end procedure
```

A sample MATLAB code used to generate initial population is given as follows:

```
function initPOP = GenerateInitialPopulation
initPOP = rand(100,1)
end
```



	Col_1	Col_2	Col_3	Col_4	Col_5	Col_6	Col_7	Col_8	Col_9	Col_10
1	0.1934	0.9508	0.8954	0.1439	0.0946	0.5230	0.5386	0.9883	0.7485	0.6228
2	0.7544	0.4976	0.5825	0.6060	0.3232	0.3253	0.9917	0.9295	0.5433	0.7966
3	0.3463	0.7551	0.5827	0.2545	0.7696	0.8318	0.7552	0.4095	0.3381	0.7459
4	0.4186	0.7424	0.8549	0.3242	0.2341	0.8103	0.9805	3.4146e-04	0.8323	0.1255
5	0.1557	0.8311	0.0349	0.4018	0.7404	0.5570	0.2348	0.5409	0.5526	0.8224
6	0.8190	0.1565	0.8854	0.4064	0.6928	0.2630	0.5286	0.2077	0.9575	0.0252
7	0.6249	0.4573	0.4077	0.3862	0.8241	0.6806	0.0514	0.2193	0.8928	0.4144
8	0.7386	0.6181	0.0364	0.6098	0.8280	0.2337	0.7569	0.3258	0.3565	0.7314
9	0.8051	0.9322	0.7461	0.1669	0.2934	0.4564	0.6020	0.0959	0.5464	0.7814
10	0.0672	0.8351	0.1548	0.1881	0.3094	0.3846	0.8572	0.7475	0.3467	0.3673

(a) Initial Population of PNN spread

	Col_1	Col_2	Col_3	Col_4	Col_5	Col_6	Col_7	Col_8	Col_9	Col_10
1	0.2175	0.4988	0.4925	0.1831	0.1238	0.3613	0.3644	0.5081	0.4456	0.3956
2	0.4488	0.3550	0.3738	0.3925	0.3050	0.3050	0.5081	0.4956	0.3644	0.4581
3	0.3050	0.4488	0.3738	0.2581	0.4519	0.4706	0.4488	0.3238	0.3050	0.4456
4	0.3300	0.4456	0.4769	0.3050	0.2425	0.4613	0.5081	0.7238	0.4706	0.1394
5	0.1956	0.4706	0.0894	0.3238	0.4456	0.3675	0.2425	0.3644	0.3644	0.4644
6	0.4613	0.1956	0.4894	0.3238	0.4331	0.2613	0.3581	0.2269	0.5019	0.0988
7	0.3956	0.3488	0.3238	0.3300	0.4644	0.4269	0.0863	0.2331	0.4925	0.3238
8	0.4456	0.3956	0.0894	0.3925	0.4675	0.2425	0.4488	0.3050	0.3081	0.4456
9	0.4613	0.4956	0.4456	0.1925	0.2831	0.3456	0.3863	0.1238	0.3644	0.4581
10	0.0863	0.4706	0.1956	0.2175	0.2956	0.3269	0.4769	0.4456	0.3050	0.3175

(b) Fitness values of initial population of PNN spread

Figure 8.2: Initial population of PNN spread (chromosomes) and their associated fitness values

2. Fitness evaluation: Each chromosome is evaluated using classification error from the PNN Classifier itself (see Algorithm 10). The actual classification error is computed from the confusion matrix generated from the PNN Classifier using the Training Set, Test Set, ClassInformation and the PNN Spread. Figure 8.2 shows 100 chromosomes (PNN spread) and their associated fitness values.

3. Selection mechanisms: The aim of selection mechanism in GA is to make sure the population (solution candidates) is being constantly improved over all fitness values. The selection mechanism helps the GA in discarding bad spread values and keeping only the best individuals. The employed scheme in this study was tournament selection of size 2, where two chromosomes are selected from the population after the Elite kids are taken out and the best of the two chromosomes, (using fitness ranking), was selected. Tournament selection was performed iteratively until the new population is filled up. Tournament scheme was

used for both GAs in chapter 5 & this chapter.

4. Genetic operators: For GA1-PNN (GA Toolbox) we used uniform mutation and heuristic crossover as genetic operators while for GA2-PNN (our implementation), gaussian mutation and arithmetic crossover were used. Both mutation operators (uniform and gaussian) perturb each chromosome by adding a random number from the appropriate or associated distribution to each parent from the tournament selection. Given that the chromosomes are x_i, i = 1 (1) Genomelength, the mutation is carried out as shown in Equation 8.1.1

$$x_i' = x_i + \sigma_i{}'.N_i(\mu, \sigma) \tag{8.1.1}$$

where $N_i(\mu, \sigma)$ represents the outcomes of random variables drawn from a Gaussian distribution having mean μ and standard deviation σ. The parameter for the GA mutation operator are shown in Table 4.1. Mutation operator increases the diversity of a population and thereby increases the likelihood that the GA will produce individuals which have better fitness values.

The heuristic crossover on the other hand, returns a child chromosome lying on the straight line containing the two parents chromosomes. It uses fitness values of the two parent chromosomes to determine the direction of the search. Thus the offsprings produced by the heuristic crossover is:

Child1 = $P_2 + RD * (P_1 - P_2)$

Child2 = P_1

where RD is a random number between 0 and 1 and P_1, P_2 are the two parent chromosomes (P_1 is the best parent, and P_2 is the worst). Arithmetic crossover also linearly combines two parent chromosomes to produce new offsprings but according to the following equations:

Child1 = αP_1 + (1-α)P_2

Child2 = (1-α)P_1 + αP_2

where α is a random number produced before the crossover operation.

Crossover operator enables the GA to extract the best genes from different individuals and recombine them into another children which are potentially superior.

5. GA stopping criteria: The two stopping criteria used for this GA are:

   (1) maximum number of generation and

   (2) number of GA iteration. These are already listed in Table 8.1.

The list of genetically optimised PNN spreads and the associated classification accuracies generated from this chapter using Algorithms 11 & 12 are shown in Figure 8.4.

Algorithm 10 Fitness Function Evaluation

```
procedure  FitFunction()
    Parameters: TrainingSet, TestSet and the ClassInformation.
    Input Spread from the GAPopulationFunction.
    PNNClassify(TrainingSet, TestSet, ClassInformation, Spread) ↦ FitnessValue.
    Output FitnessValue.
end procedure
```

Algorithm 11 Pseudocode for GAPNNSpreadOptimizer

```
procedure GAPNNSpreadOptimizer(PNNParameters, GAParameters)
    Generate Initial population of PNN Spread.
    Set GA()Parameters as shown in Table 8.1.
    Set Counter = 1
    Do
    Simulate GA()
    Output best chromosome ↦ Spread and store its value in a variable ListOfChromosomes.
    Counter = Counter + 1
    Until Counter = M, where M = Number of GA simulation needed.
    Output ↦ ListOfChromosomes.
end procedure
```

Algorithm 12 Pseudocode for PNNAccuracy versus PNNSpread Plot

```
procedure Performance(TrainingSet,TestSet,ClassInformation,ListOfChromosomes)
    Input: TrainingSet, TestSet, ClassInformation, ListOfChromosomes.
    Set M = length(ListofChromosomes).
    Set Counter = 1
    Do
    spread = ListOfChromosomes(Counter)
    Accuracy(Counter) = PNNClassify(TrainingSet, TestSet, ClassInformation, spread)
    Counter = Counter + 1
    Until Counter = M.
    Graph ↦ Plot(ListOfChromosomes, Accuracy).
end procedure
```

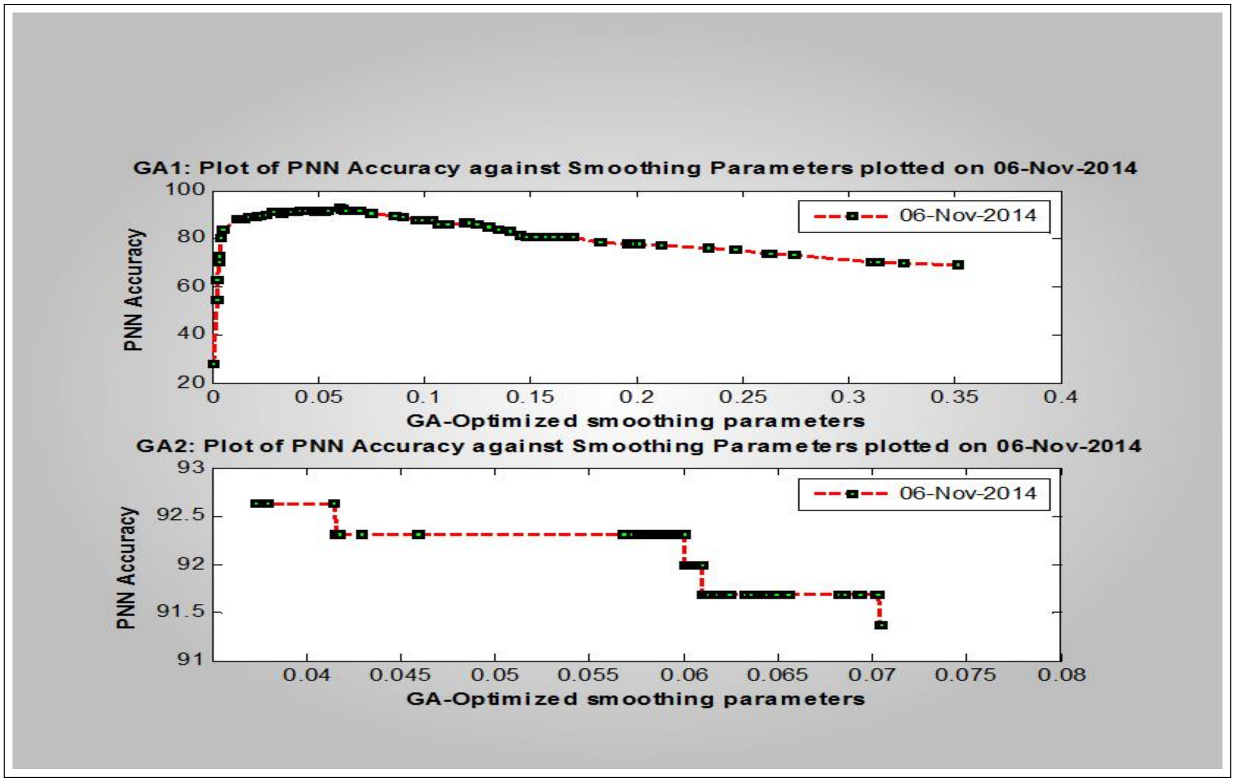
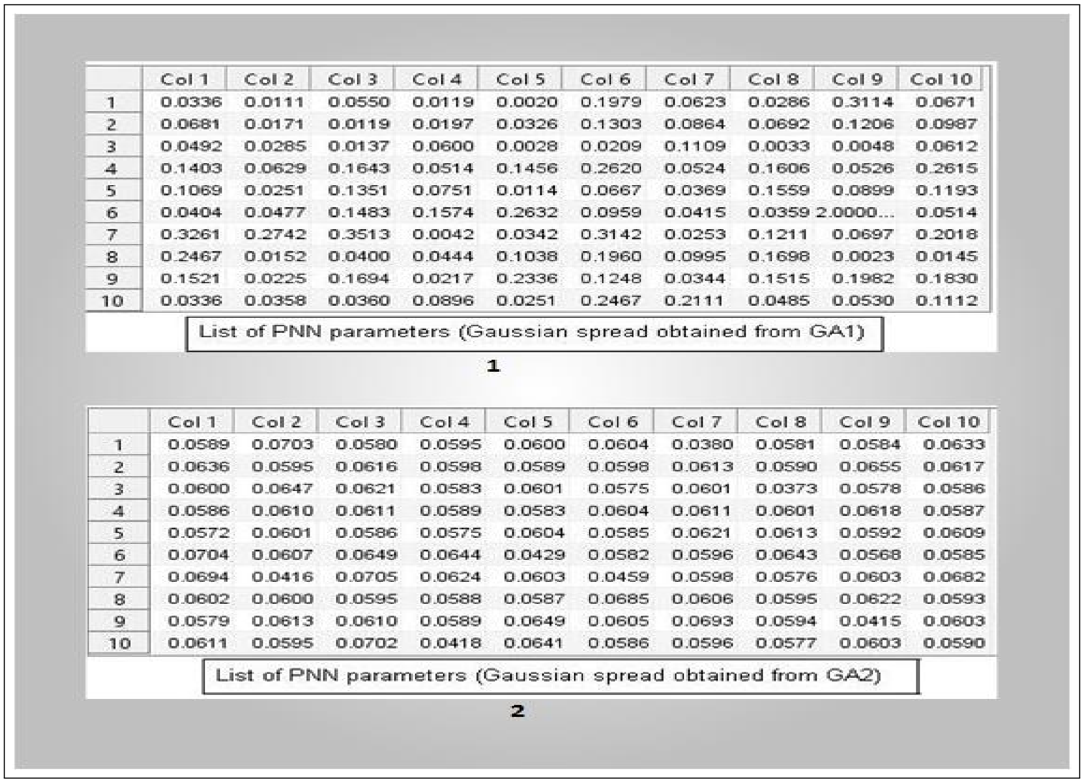


Figure 8.3: Variation of PNN Accuracy with GA-Based Smoothing Parameters



	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10
1	0.0336	0.0111	0.0550	0.0119	0.0020	0.1979	0.0623	0.0286	0.3114	0.0671
2	0.0681	0.0171	0.0119	0.0197	0.0326	0.1303	0.0864	0.0692	0.1206	0.0987
3	0.0492	0.0285	0.0137	0.0600	0.0028	0.0209	0.1109	0.0033	0.0048	0.0612
4	0.1403	0.0629	0.1643	0.0514	0.1456	0.2620	0.0524	0.1606	0.0526	0.2615
5	0.1069	0.0251	0.1351	0.0751	0.0114	0.0667	0.0369	0.1559	0.0899	0.1193
6	0.0404	0.0477	0.1483	0.1574	0.2632	0.0959	0.0415	0.0359	2.0000...	0.0514
7	0.3261	0.2742	0.3513	0.0042	0.0342	0.3142	0.0253	0.1211	0.0697	0.2018
8	0.2467	0.0152	0.0400	0.0444	0.1038	0.1960	0.0995	0.1698	0.0023	0.0145
9	0.1521	0.0225	0.1694	0.0217	0.2336	0.1248	0.0344	0.1515	0.1982	0.1830
10	0.0336	0.0358	0.0360	0.0896	0.0251	0.2467	0.2111	0.0485	0.0530	0.1112

List of PNN parameters (Gaussian spread obtained from GA1)

1

	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10
1	0.0589	0.0703	0.0580	0.0595	0.0600	0.0604	0.0380	0.0581	0.0584	0.0633
2	0.0636	0.0595	0.0616	0.0598	0.0589	0.0598	0.0613	0.0590	0.0655	0.0617
3	0.0600	0.0647	0.0621	0.0583	0.0601	0.0575	0.0601	0.0373	0.0578	0.0586
4	0.0586	0.0610	0.0611	0.0589	0.0583	0.0604	0.0611	0.0601	0.0618	0.0587
5	0.0572	0.0601	0.0586	0.0575	0.0604	0.0585	0.0621	0.0613	0.0592	0.0609
6	0.0704	0.0607	0.0649	0.0644	0.0429	0.0582	0.0596	0.0643	0.0568	0.0585
7	0.0694	0.0416	0.0705	0.0624	0.0603	0.0459	0.0598	0.0576	0.0603	0.0682
8	0.0602	0.0600	0.0595	0.0588	0.0587	0.0685	0.0606	0.0595	0.0622	0.0593
9	0.0579	0.0613	0.0610	0.0589	0.0649	0.0605	0.0693	0.0594	0.0415	0.0603
10	0.0611	0.0595	0.0702	0.0418	0.0641	0.0586	0.0596	0.0577	0.0603	0.0590

List of PNN parameters (Gaussian spread obtained from GA2)

2

Figure 8.4: List of PNN parameters (Gaussian spread) generated by the two Genetic Algorithms

8.1.1 Design of Image Classification System

The steps involved in the design of the GA-PNN image classification system has been shown in Figure 7.8 are further described in Figure 8.5. These steps are similar to other image classification systems which have been reported in this study. The novelty of this study is seen in the inclusion of genetic image segmentation module and the optimization module for the PNN classifier which incorporated GA for optimization. The entire classification system (including the GUI) was implemented using MATLAB 2013a.

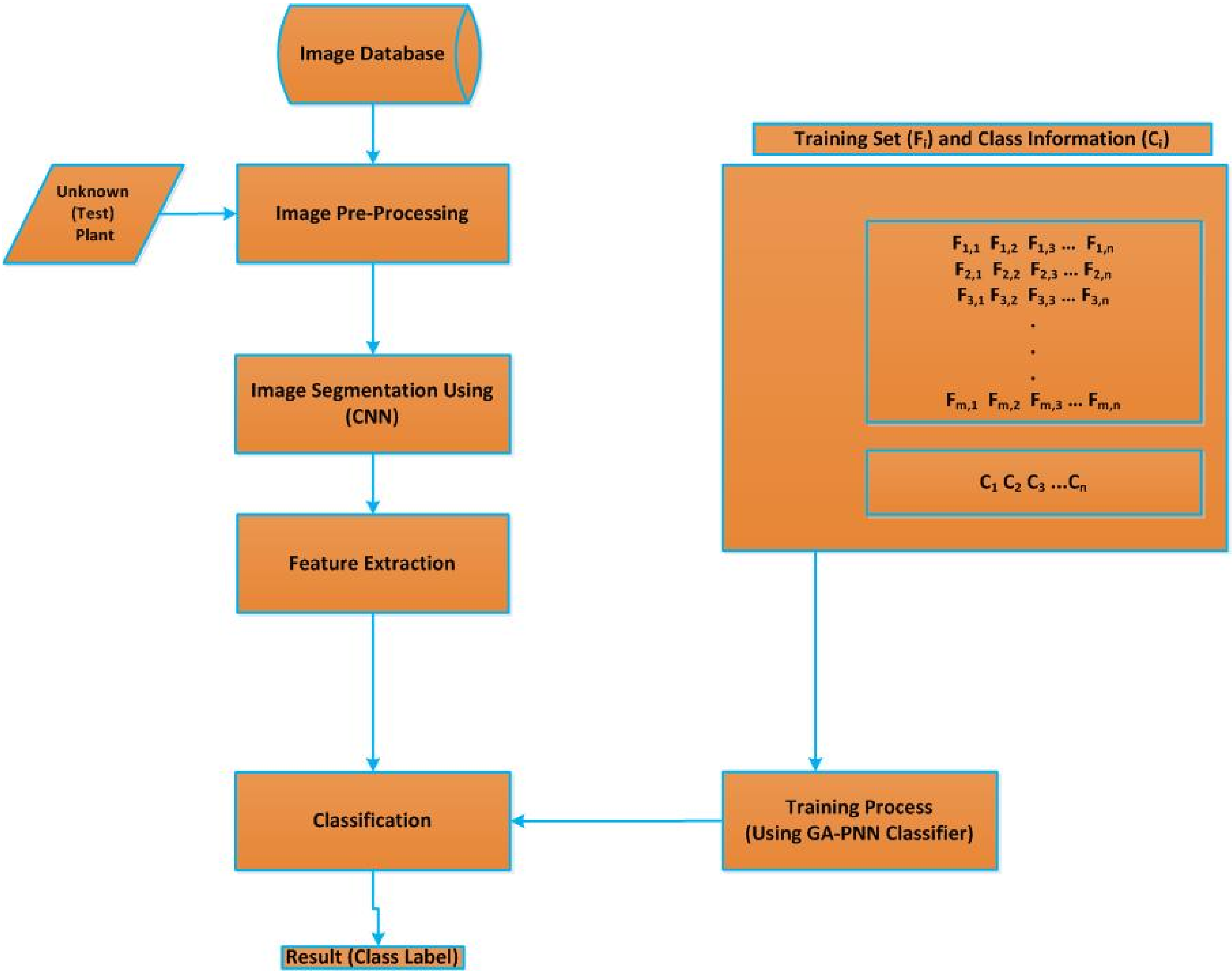


Image Database
Unknown (Test) Plant
Image Pre-Processing
Image Segmentation Using (CNN)
Feature Extraction
Classification
Result (Class Label)
Training Set (F_i) and Class Information (C_i)
$F_{1,1}$ $F_{1,2}$ $F_{1,3}$ … $F_{1,n}$
$F_{2,1}$ $F_{2,2}$ $F_{2,3}$ … $F_{2,n}$
$F_{3,1}$ $F_{3,2}$ $F_{3,3}$ … $F_{3,n}$
$F_{m,1}$ $F_{m,2}$ $F_{m,3}$ … $F_{m,n}$
C_1 C_2 C_3 …C_n
Training Process (Using GA-PNN Classifier)

Figure 8.5: Learning system based on PNN Classifier

8.2 Results

Two GAs were used for comparison in this study including the standard MATLAB GA Toolbox and a modified implementation used for this study.

1. MATLAB GA Toolbox (1st GA): For running the GA Toolbox, these are parameter configuration used are shown in Table 8.1 . In the this approach the GA involve runs once and the final population is used as final list of chromosomes. The best PNN spread (associated with the maximum accuracy) for this approach is 0.035. With inclusion of colour features, the new spread obtained was 0.0371 which increased the accuracy of the PNN classifier from 91.06 % to 91.37 %

2. This study GA Implementation (2nd GA):. The parameter setting for this GA is shown in Table 8.2. In this GA, and in a similar approach to the first approach for the GA Toolbox, the chromosomes are ranked based on their fitness values. The positional index of the best fitness value is used to obtain the best chromosome in each generation. At the end of 100 generation, a list of chromosomes is then obtained. The best chromosome (PNN spread) for the 2nd GA was 0.060 with associated accuracy of 92.62%. A similar optimization by PSO produced spread of 0.0336 which confirms the choice of the spread to be used for performance improvement. The accuracy for PSO was 90.24 % which is slighly less than that of GA.

8.2.1 Comparison of PNN with some other classifiers

The accuracies of the genetic PNN was compared with a number of other known classifiers (already described in subsections 16a, 16c, 16d, 16e & 16f of section 16). The results are shown in Figure 8.6 with associated ranking displayed at the top right corner. The performance of both genetic PNN and ensemble methods are nearly the same but genetic PNN still outperformed the ensemble methods. The third position in ranking was exhibited by MLP, followed by kNN and NBC for 4th and 5th rankings respectively. The PNN was made to be adaptively robust compared to other classifiers. A hybrid model CNN-PNN-GA is envisioned as a good choice for computer-based plant recognition since it's able to address some of the problems facing computer-based vision systems for plant idenification. Both CNN and PNN are classifiers and hence, hybrid model.

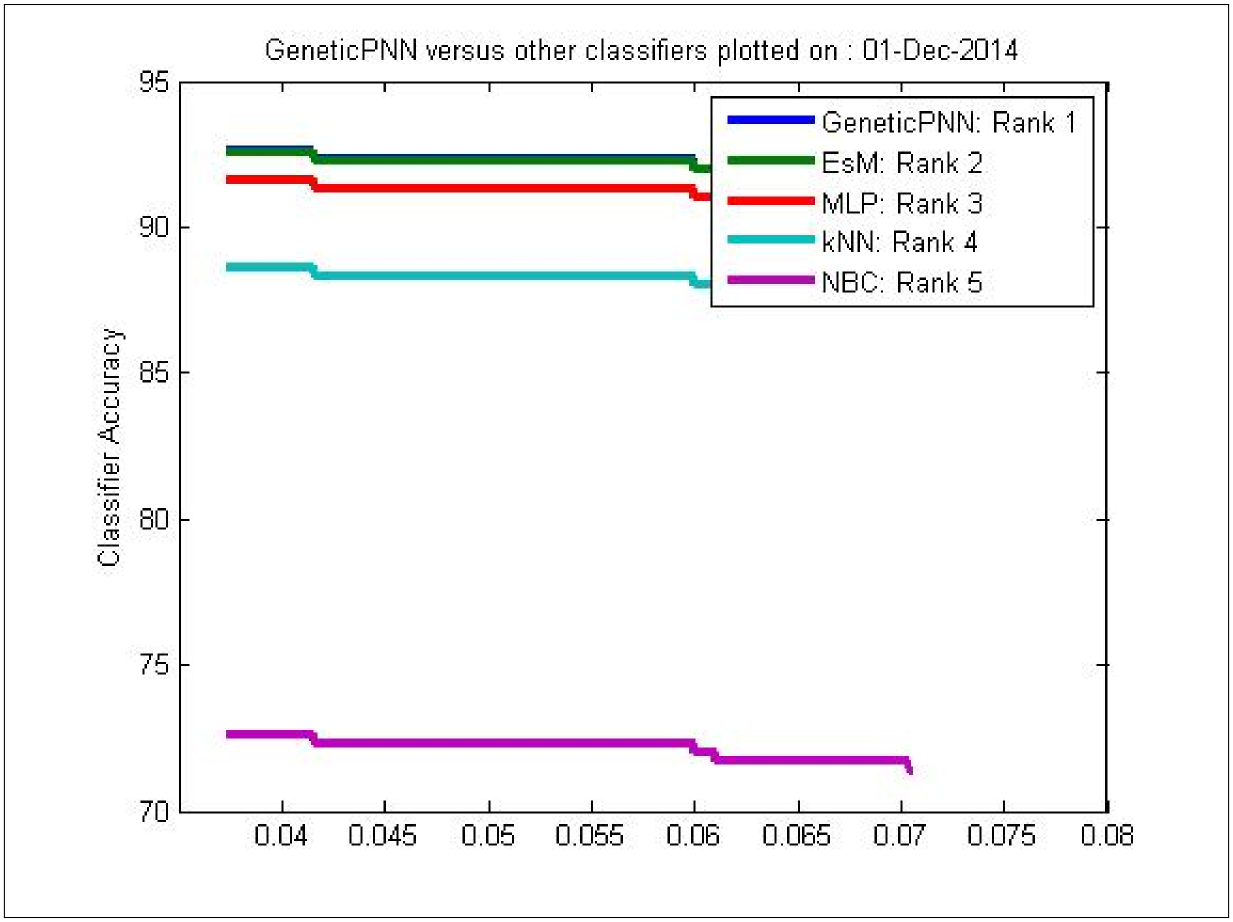


Figure 8.6: Accuracies comparison on Genetic PNN with some other classifiers

8.2.2 Results and Discussion

The plot of PNN accuracy against the PNN spread for the two GA approaches are shown in Figure 8.3. It was found for this study that the best PNN accuracy was achieved from the 2nd GA. It was found that the GAs provided the best PNN spread without resulting in PNN under and over fitting. The peak accuracy for the two plots shown in this figure were based on the spread value of 0.035 and 0.060 for 1st GA and 2nd GA respectively. With the use of GA optimization, the performance of the PNN was improved. The GA searched for all wide range of possible real numbers to represent the PNN spread and brought a candidate solution for optimal classification accuracy of the PNN.

Chapter 9

General discussion and Conclusions

9.1 Introduction

In this last chapter, the overview of the research activities, results and discussions are provided. In order to achieve this, the research questions from Chapter 1 are provided and aligned with each phases of the research methodology.

The Main Research Question eluded from the research was:
How can a hybrid-based approach based on (CNN, PNN, and GA) be employed for plant leaves classification systems?

The response to this question has been provided in Chapters 3, 4, 5, 6, 7, & 8

1. Application of CNN
   CNN has been used in this work to extract optimal edge pixels from the images of plant leaves found in the Flavia dataset. Figures 3.2, 4.5, & matrix templates in Equation 4.1.2 show the outputs derived after optimizing the CNN templates with GA. Edge pixels were needed by some of the features extracted from the images. For example, Fourier Descriptors (FD) from Chapter 4 rely heavily on edge points as it's a shape-based descriptors. CNN was used to obtain optimal edge images before being passed over to FD module for final feature extraction. It was established and concluded that CNN-based edge pixels outperformed the

conventional edge operators such as Canny, Sobel, LoG, etc, in terms of computational time and classification accuracies.

2. Application of PNN

   The probabilistic neural network (PNN) was the main classifier used in this study. The detailed description of PNN and its relation to Naive Bayes Classifier (NBC) has been discussed in Chapter 7. The mathematical nature (computational and theoretical properties) of the PNN was also given in Chapter 7.

3. Application of GA

   GA has been heavilly applied throughout this thesis as it was used for both feature selection and optimization of parameters of the concerned learning machines.

Sub-Research Question 1

What are the suitable techniques for image segmentation for feature extraction in plant leaf classification systems?

This response to this subquestion has been provided in Chapter 3, which details the numerical method used in discretizing the CNN (image edge detector model) and the type of activation function used during the simulation process (see section 3.3.3.2). The activation function used is shown in Equation 3.3.24. The CNN templates are the matrix coefficients for the systems of equations in 3.3.23 & 3.3.24. The numerical methods (Runge-Kutta (R-K)) used in descritizing the ODE (CNN) shown in Equation 3.3.23 are given as Equations 3.3.28,3.3.29,3.3.30, 3.3.31, 3.3.32, 3.3.33. The use of R-K method reduced greatly the descritization errors of the concerned ODE. This is the preferred method by researchers in image processing.

Sub-Research Question 2

How can GA be used to obtain suitable set of plant leaf features (shape, colour, and texture) associated with plant leaf recognition?

The response to this subquestion has been provided in Chapter 5.

Feature extraction and selection are important part of this study as many features were extracted and combined in a single study to eliminate some of the problems facing existing systems on plant leaves identification. These problems include misclassification of species. The rational for using several feature is to be able to have them in a database and then use a dimensionality reduction techniques such as GA, PSO and PCA on them to fetch out the best features so as to have optimal accuracy from the underlying learning machine models.

Sub-Research Question 3

How can effectiveness of GA compared to PSO, and PCA be established for feature selection?

The response to this subquestion has been provided in Chapter 6. The features selected by both GA and PSO were 14 in number. For this reason, PCA was made to selected 14 PCs. The three feature sets based on GA, PSO and PCA were tested on the concerned classifier (PNN) with GA show the best classification results (see Figure 6.17). A result from using Mutual Information (MI) as the fitness function instead of kNN shows that 85% similarities.

Sub-Research Question 4

How can PNN-based classification of leaves be optimized through the use of GA techniques?

The response to this subquestion has been provided in Chapter 8. The parameters of both CNN and PNN used were optimized using a GA so as to have optimal accuracy in the image classification system. The final matrix templates generated by applying GA on CNN was shown in Equation 4.1.2 while the best PNN spread (gaussian smoothing parameter) of the underlying training set was genetically computed as 0.035 which agrees with that fact that the PNN spread value should always be fixed around $\frac{1}{c} \approx 0.031$, where $c = 32$ is the number of classes in the dataset.

9.2 Overview of experimental phases

The experimental phases (re-iterated and re-shown in Figure 9.1) comprised of the following steps:

1. Image acquisition: The images used in this work are standard images of plant species provided freely by Wu et al. (2007).

2. Image pre-processing: The images are pre-processed herein for use by the image segmentation modules involving the use of Cellular Neural Networks, and other ROI operators such as Canny, Sobel, Prewitt, and LoG. The matrix templates assoaciated with these edge operators are shown in Equations 3.3.15, 3.3.16, 3.3.17, & 3.3.18. Pre-processing is very crucial for non-colour and sometimes, color-based features. For this study the image pre-processing includes, image resizing, color-to-gray conversion and then gray-to-binary conversion. The whole stages involved in the image pre-processing module were iteratively done in batch mode as there were nearly 2000 images in the database.

3. Image segmentation

   The segmentation modules in this work was done via thresholding and edge detection. The edge detection was particulary carried out using genetically optimized CNN cloning templates and compared with the coventional edge operators such as Canny, Sobel, LoG, and Robert. The CNN improved the operational speed of the image classification system. The GA enabled the CNN to bring out detailed edge points from the images and thus contributed to the efficiency of the entire system. In the field of agricultural informatics or precision agriculture, the use of genetic CNN is a novel application.

4. Feature extraction

   This work provides a myriad of image descriptors (extracted features). The features were discussed in Chapter 4 and re-listed here for reference purpose. The image descriptors used are Zernike Moments (ZM), Fourier Descriptors (FD), Lengendre Moments (LM), Hu 7 Moments (Hu7M), Texture Properties (TP) , Geometrical Properties (GP), and Colour features (CF). The total number of these features were 112 as shown in Table 4.3. Only 12 out of these features were colour-based. The remaining 100 features were non-colour. The rational for providing both non-colour and colour features dataset is to be able to explore the second PhD dataset which comprises only binary images of 100 plant species. The combination of all these features in a single study like this is a novel, tremendous and useful approach as most researchers use either a few or combination of few of them. The rational for using as many of these as possible is to make sure the problems statements associated with this study is answered. The original feature set for this study is thus a 1907×112 and 1907×100 matrices of real numbers for both color and non-color features respectively.

5. Feature selection and analysis

   The 1907×112 feature space was further reduced by GA, PSO, and PCA. The populations from both evolutionary algorithms were evaluated using the same fitness function as shown in Equation 5.4.6. The features selected by the GA gave the best classification accuracy (See Figure 6.17). One of the papers ((O. Babatunde, Armstrong, Leng, & Diepeveen, 2015c)) published from this work demonstrates the impact of both GA and PSO on some selected classifiers. The original features space was reduced from a 1907 x 112 matrix of real numbers to 1907 x 14 matrix of real numbers. In other words, both GA and PSO selected only 12.50% of the original dataset. Nine different classification models were tested as shown in the results Table of the paper . The results herein showed that both GA and PSO-based features outperformed the classification models based on the original features while GA-based feature in turn outperformed the PSO-based features. The features selected by both GA and PSO are somewhat similar as shown in Figure 6.5. This may be due to the same fitness function used for both. However the different nature of the two evolutionary algorithms made sure there are only 57% similarly between the numbers of features selected by both.

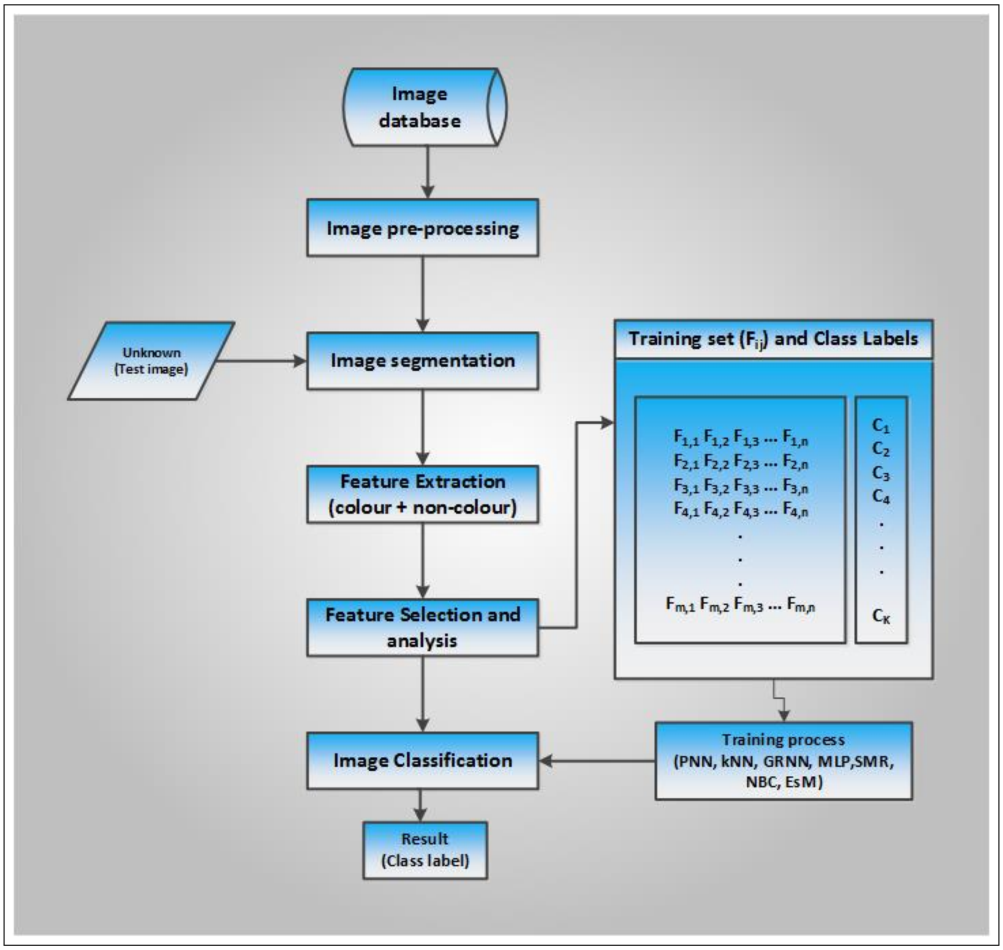


Image database
Image pre-processing
Unknown (Test image)
Image segmentation
Feature Extraction (colour + non-colour)
Feature Selection and analysis
Training set (F_{ij}) and Class Labels
$F_{1,1}$ $F_{1,2}$ $F_{1,3}$ … $F_{1,n}$
$F_{2,1}$ $F_{2,2}$ $F_{2,3}$ … $F_{2,n}$
$F_{3,1}$ $F_{3,2}$ $F_{3,3}$ … $F_{3,n}$
$F_{4,1}$ $F_{4,2}$ $F_{4,3}$ … $F_{4,n}$
$F_{m,1}$ $F_{m,2}$ $F_{m,3}$ … $F_{m,n}$
C_1
C_2
C_3
C_4
C_K
Training process (PNN, kNN, GRNN, MLP,SMR, NBC, EsM)
Image Classification
Result (Class label)

Figure 9.1: General overview of PhD work: The works in this study were based on amalgamation of several features and both genetic segmentation and classification techniques. The GA-based feature selection (wrapper method) part of this work proved very useful as it enabled the classifier to be more accurate. The work as seen in this figure is easily adaptible to forensic application by changing only the images in the database and or little amendment on the pre-processing and segmentation module.

The features selected by the GA were:

- ZMI(5,1): Zernike moment of order 5 and repetition 1.
- ZMI(6,0): Zernike moment of order 6 and repetition 0.
- ZMI(6,4): Zernike moment of order 6 and repetition 4.
- LengM (Im, 1): Legendre moment of order 1.
- Entropy(Im): Image entropy.
- Solidity (Im): Image solidity.
- Leaf Major axis
- Leaf Perimeter
- MeanR: Mean of red band of RGB image.
- MeanG: Mean of green band of RGB image.
- MeanB: Mean of blue band of RGB image.
- sdR: Standard deviation of red band of RGB image.
- sdG: Standard deviation of green band of RGB image.
- sdB: Standard deviation of blue band of RGB image.

The features selected by the PSO were:

- ZMI(4,2): Zernike moment of order 4 and repetition 2.
- ZMI(5,3): Zernike moment of order 5 and repetition 3.
- ZMI(6,4): Zernike moment of order 6 and repetition 4.
- ZMI(9,5): Zernike moment of order 9 and repetition 5.

- ZMI(9,7): Zernike moment of order 9 and repetition 7.
- Solidity (Im): Image solidity.
- Leaf Perimeter
- Leaf minor axis
- 21st coefficient of Fourier Descriptors
- MeanR: Mean of red band of RGB image.
- MeanG: Mean of green band of RGB image.
- MeanB: Mean of blue band of RGB image.
- sdR: Standard deviation of red band of RGB image.
- sdG: Standard deviation of green band of RGB image.

Features selected by Multi-Objective Genetic Algorithm (MOGA) were 8 in number and they are:

- ZMI(8,2): Zernike moment of order 8 and repetition 2.
- ZMI(9,3): Zernike moment of order 9 and repetition 3.
- Leaf minor axis
- Leaf major axis
- MeanR: Mean of red band of RGB image.
- MeanG: Mean of green band of RGB image.
- MeanB: Mean of blue band of RGB image.

The implication here is that the features common to both PSO and GA should be looked at when considering building a computer-based vision systems for identifying plant species and or other image recognition tasks. Looking at the features selected by the MOGA, it can be seen that 5 of these were also selected by both GA and PSO. These five features can also be used to construct image classification model but it does not include any Zernike moment and thus, was not prefered above PSO and GA. The ZM used in this study was made to be invariant to translation, rotation and scaling (TRS). It is considered beneficial to have such descriptors in an ideal computer-based vision systems to allow varities in image capture.

6. Image Classification: The image classification was done via genetically optimized PNN. The optimization of PNN parameter using GA improved the classification accuracy of the learning system. As the tuning parameter (spread) for the PNN cannot be chosen just arbitrarily if optimal accuracy is desired, the GA was used to bring out the spread value which eventually improved the performance of the PNN. The value of the PNN spread is fully dependent on the underlying training sample used to train the PNN. In comparison to other common classifiers, the performance of the genetically optimized PNN was the best in terms of classification accuracy. The results herein shows that PNN, when optimized by an evolutionary algorithm, can be used for almost any kind of pattern recognition. The PNN of course, is a multi-class learning machine and can thus be used in wide varities of pattern classification. Both color and non-color features were provided in this study to capture a wide varities of descriptors from the given set of images. This will make this work to adaptible for either color-based vision system , or non-color based vision system or both.

9.3 Contribution to knowledge

The ideas in this work are considered to be novel in image classification systems. The methods used in this research are very adaptive and could be used for other classification systems. The use of several image descriptors in one study is a novel approach as most of the existing works on computer-based vision systems for plant species identification have used only a few image descriptors. The feature-selection and analysis part of this study were also novel as the GA was built to be very selective. The novel fitness function (see Equation 5.4.6) used by both GA and PSO was the main driving force in the feature selection module. This work can also be adapted and used for forensic purposes, general pattern recognition, and data minning. This work also forsters a strong link between mathematics and image processing as one of the edge detector used was a class of ordinary differential equation (ODE).

9.4 Conclusion and future directions

This research involved application of PNN, CNN, GA and features derived from the images of the plants. The features selected by GA proved more effective than those selected by PSO and PCA. The selected features were further analysed statistically using ANOVA and correlation techniques. The essense of the statistical test was to ensure that the features selected by both GA and PSO were void of redundancy. The results of the analysis show that the developed feature selection techniques were effective in bringing out combinatorial set of features that improved performance of learning machine and also not redundant. The computational and theoretic properties of PNN were also presented. The reason for including these properties of PNN was to be able to study abstract nature of the PNN with respect to classification. This may open up new waves of opportunities for future researchers to study PNN in more depth. In regard to the computer-based vision system developed for plant species identification, future works lie in the use of more organs of plants such as flowers and fruits to complement features derived from the use of only leaf images. A combination of hybrid or possibly more descriminative classifier may be a possible solution. Several classifiers have already been developed. However, it is expected that newly discovered ones may be useful for this purpose. This work could also be more accessible through deployment on mobile devices which could assist field botanists. As the number of images used in this study were based on just 32 species of plants, more images and species added to the database would further improve the global application of this kind of work.

APPENDIX

Appendix I
Definitions of terms in Image Processing

1. Image: An image is defined as a 2D light intensity function , say, $f(x,y)$, where x and y are the spatial coordinates and f denotes the brightness or gray level at the point (x,y). If the image is generated from a physical scenario, then the intensity values of such image are proportional to energy radiated by the physical source. Therefore, $f(x,y)$ is always assumed to be nonzero and finite as given by the inequality $0 < f(x,y) < \infty$.

2. Digital Image: A digital image is a matrix of pixels representing image intensities (set of integers $\{0,1,2,...,L,L>1\}$) at spatial coordinates. All digital images are assumed to have been discretized in both spatial coordinates and brightness. In common practice, L = 256 and each pixel value is stored in one byte. If $L=2$, the image is called binary image.

3. Sampling and Quantization: Conversion of a continous image $f(x,y)$ into a digital image is done through sampling (spatial discretization) followed by quantization (graylevel quantization).

4. Bounding box: The bounding box for an image is defined as the smallest rectangle which encloses the image. The minimum area of such bounding box is given as:

   $$A_b = L * W$$

   where :

   (a) A_b = AreaBoundingbox

   (b) L = majorAxisLength

(c) W = minorAxisLength

5. Curvature: This is defined as the rate of change of a slope. The curvature of the boundary at $p = (x_i, y_i)$ can be estimated from the change in the slope. It's given as:

$$\kappa(p) = \tan^{-1}\left(\frac{y_{i+k} - y_i}{x_{i+k} - x_i}\right) - \tan^{-1}\left(\frac{y_i - y_{i-k}}{x_i - x_{i-k}}\right)(mod 2\pi)$$

6. Bending energy: This is an image descriptor that is obtained by integrating the squared curvature $\kappa(p)$ through the boundary length L. It's a robust shape descriptor and can be used for matching shapes. The bending energy is defined in the following equation:

$$E_c = \frac{1}{L}\sum_{p=1}^{L}\kappa(p)^2, \frac{2\pi}{R} \leq E_c \leq \infty.$$

7. PCA Simplified

(a) Let X = $\{x_1, x_2, x_3, ..., x_n\}$ be a feature vector with observations in d-dimensional space R$^{\mathrm{d}}$

(b) Compute the mean $\bar{x}$ which is

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{9.4.1}$$

(c) Compute the covariance matrixs H_x as

$$H_x = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T \tag{9.4.2}$$

(d) Compute the eigenvalues , eigenvectors λ_i, v_i of H_x viz

$$|H_x - \lambda I| = 0 \tag{9.4.3}$$

(e) Arrange the eigenvectors in descending order based on their eigenvalues. The k

principal components are the eigenvectors corresponding to the k largest eigenvalues.

(f) For any observed vector x, the k principal components are given by

$$y = W^T(x - \bar{x}) \tag{9.4.4}$$

where

W = $[v_1, v_2, v_3, ..., v_k]$

The observed vector x can be reconstructed from the PCA basis as

$$x = Wy + \bar{x} \tag{9.4.5}$$

8. Convex set: A set $\mathbb{X}$ is said to be convex if for every $x, y \in \mathbb{X}, \exists\, x, y \in \mathbb{X} : \lambda x + (1-\lambda)y \in \mathbb{X}, \lambda \in [0,1]$

9. Zernike functions, denoted $Z_n{}^m(r, \theta)$, are infinite set of orthogonal functions defined on the unit circle $r \in [0,1], \theta \in [0, 2\pi]$

10. Rotational invariance of Zernike Moments (ZM) Using the polar coordinates as shown in Chapter 4, the rotational invariance properties of ZM can easily be expressed as follows: Let

$$f'(\rho, \theta) = f(\rho, \theta - \alpha) \tag{9.4.6}$$

Then

$$Z_{nm} = \left[\frac{n+1}{\pi}\right] \int_0^{\pi} \int_0^1 f(\rho, \theta) R_{nm}(\rho) \exp(-im\theta) \rho d\rho d\theta \tag{9.4.7}$$

By inserting the rotated image into Equation 9.4.7, then we have

$$Z'_{nm} = \left[\frac{n+1}{\pi}\right] \int_0^{\pi} \int_0^1 f(\rho, \theta - \alpha) R_{nm}(\rho) \exp(-im\theta) \rho d\rho d\theta \tag{9.4.8}$$

and letting $\theta_1 = \theta - \alpha$, we have

$$Z'_{nm} = \left[\frac{n+1}{\pi}\right] \int_0^{\pi} \int_0^1 f(\rho,\theta_1) R_{nm}(\rho) \exp(-im(\theta_1 + \alpha))\rho d\rho d\theta \tag{9.4.9}$$

$$Z'_{nm} = \left[\frac{n+1}{\pi}\right] \int_0^{\pi} \int_0^1 f(\rho,\theta_1) R_{nm}(\rho) \exp(-im(\theta_1)\rho d\rho d\theta . [\exp(-im(\alpha))] \tag{9.4.10}$$

$$Z'_{nm} = Z_{nm}[\exp(-im(\alpha))] \tag{9.4.11}$$

$$Z'_{nm} = Z_{nm}[\cos(m\alpha) - i\sin(m\alpha)] \tag{9.4.12}$$

$$Z'_{nm} = Z_{nm} \tag{9.4.13}$$

11. On Zenike polynomials: The Zernike polynomials also satisfy the following recurrence relation depends neither on the degree nor on the azimuthal order of the radial polynomials

$$R_n^m(\rho) + R_{n-2}^m(\rho) = \rho \left[R_{n-1}^{|m-1|}(\rho) + R_{n-1}^{m+1}(\rho)\right] \tag{9.4.14}$$

"

Table 9.1: Zernike Polynomials

S/N	Representation	Formular
1	$R_{0,0}(r)$	1
2	$R_{1,1}(r)$	r
3	$R_{2,0}(r)$	$2r^2 - 1$
4	$R_{2,2}(r)$	r^2
5	$R_{3,1}(r)$	$3r^3 - 2r$
6	$R_{3,3}(r)$	r^3

7	$R_{4,0}(r)$	$6r^4 - 6r^2 + 1$
8	$R_{4,2}(r)$	$4r^4 - 3r^2$
9	$R_{4,4}(r)$	r^4
10	$R_{5,1}(r)$	$10r^5 - 12r^3 + 3r$
11	$R_{5,3}(r)$	$5r^5 - 4r^3$
12	$R_{5,5}(r)$	r^5
13	$R_{6,0}(r)$	$20r^6 - 30r^4 + 12r^2 - 1$
14	$R_{6,2}(r)$	$15r^6 - 20r^4 + 6r^2$
15	$R_{6,4}(r)$	$6r^6 - 5r^4$
16	$R_{6,6}(r)$	r^6
17	$R_{7,1}(r)$	$35r^7 - 60r^5 + 30r^3 - 4r$
18	$R_{7,3}(r)$	$21r^7 - 30r^5 + 10r^3$
19	$R_{7,5}(r)$	$7r^7 - 6r^5 3$
20	$R_{7,7}(r)$	r^7
21	$R_{8,0}(r)$	$70r^8 - 140r^6 + 90r^4 - 20r^2 + 1$
22	$R_{8,2}(r)$	$56r^8 - 105r^6 + 60r^4 - 10r^2$
23	$R_{9,1}(r)$	$126r^9 - 280r^7 + 210r^5 - 60r^3 + 5r$
24	$R_{9,3}(r)$	$84r^9 - 168r^7 - 105r^5 - 20r^3$
25	$R_{9,5}(r)$	$36r^9 - 56r^7 + 21r^5$
26	$R_{9,7}(r)$	$9r^9 - 8r^7$
27	$R_{9,9}(r)$	r^9

12. Legendre Functions:

According to (Abramowitz & Stegun, 1965), the Legendre moments are associated with Legendre equation given by:

$$\frac{d}{dx}\left[(1-x^2)\frac{dy}{dx}\right] + \left[n(n+1) - \frac{m^2}{1-x^2}\right] y = 0 \tag{9.4.15}$$

The solutions ${p^m}_n$ to equation 9.4.15 are called Legendre polynomials, some of which are shown in equations 9.4.16 to 9.4.22.

$$P_0(x) = 1 \tag{9.4.16}$$

$$P_1(x) = x \tag{9.4.17}$$

$$P_2(x) = \frac{1}{2}(3x^2 - 1) \tag{9.4.18}$$

$$P_3(x) = \frac{1}{2}(5x^3 - 3x) \tag{9.4.19}$$

$$P_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3) \tag{9.4.20}$$

$$P_5(x) = \frac{1}{8}(63x^5 - 70x^3 + 15x) \tag{9.4.21}$$

$$P_6(x) = \frac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5) \tag{9.4.22}$$

13. Fourier series and transforms:

Suppose that $f(x)$ is defined on an interval $-L \leq x \leq L$, the Fourier series of $f(x)$ on $[-L, L]$ is defined to be the series

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n cos(\frac{2\pi nx}{L}) + b_n sin(\frac{2\pi nx}{L})\right) \tag{9.4.23}$$

where

$$a_0 = \frac{1}{L}\int_{-L}^{L} f(x)dx, \tag{9.4.24}$$

$$a_n = \frac{1}{L}\int_{-L}^{L} cos(\frac{2\pi nx}{L}) f(x)dx, \qquad for \qquad n = 1, 2, 3, ... \tag{9.4.25}$$

$$b_n = \frac{1}{L}\int_{-L}^{L} sin(\frac{2\pi nx}{L}) f(x)dx, \qquad for \qquad n = 1, 2, 3, ... \tag{9.4.26}$$

The numbers $a_0, a_1, a_2, a_3, ..., b_1, b_2, b_3, ...$ are the Fourier coefficients of $f(x)$ on $[-L, L]$, where n represents the rank of the harmonics while the magnitude and phase of the given harmonic component are given respectively by Equations (9.4.27) and (9.4.28).

$$FS_{abs} = \sqrt{a_n^2 + b_n^2} \tag{9.4.27}$$

$$PhaseAngle = \tan\left(\frac{a_n}{b_n}\right) \tag{9.4.28}$$

Example 1:

Let $f(x) = 2x+1, for \quad -3 \leq x \leq 3$. Let $L = 3$. Thus the Fourier Coefficients are

$$a_0 = \frac{1}{3}\int_{-3}^{3}(2x+1)dx = 2 \tag{9.4.29}$$

$$a_n = \frac{1}{3}\int_{-3}^{3}(2x+1)cos\left(\frac{2\pi nx}{3}\right) = 0, \qquad for \qquad n = 1,2,3,... \tag{9.4.30}$$

and

$$b_n = \frac{1}{3}\int_{-3}^{3}(2x+1)sin\left(\frac{2\pi nx}{3}\right) = \frac{12}{n\pi}cos(n\pi), \qquad for \qquad n = 1,2,3,... \tag{9.4.31}$$

The Fourier Series of $(2x+1)$ on $[-3,3]$ is therefore

$$1 + \frac{-12}{n\pi}\sum_{n=1}^{\infty}cos(n\pi)sin\left(\frac{\pi nx}{3}\right) = 1 + \frac{-12}{\pi}\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{n}sin\left(\frac{\pi nx}{3}\right) \tag{9.4.32}$$

The Fourier Transform(FT) of $f(x)$ can be defined as

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi ux}dx \tag{9.4.33}$$

where $i = \sqrt{-1}$, i.e i is a complex number, u is a frequency variable. Using Euler equation, $F(u)$ can be written as

$$F(u) = \int_{-\infty}^{\infty} f(x)(cos2\pi ux - isin2\pi ux)dx \tag{9.4.34}$$

The main feature of the Fourier expansion is that it defines orthogonal basis, which implies that

$$\int_{0}^{L} f_i(x) f_j(x) dx = 0, \ for \ i \neq j. \tag{9.4.35}$$

Equation 9.4.35 ensures that redundant information are eliminated in the expansion and also enforces computational simplicity (Nixon & Aguado, 2012). Peters (2011) defined the FT as the decomposition of a nonperiodic signal into (say, a $\lambda - periodic$) continous aggregation of sinusoids. This decomposition is expressed as

$$f(t) = A_0 + \sum_{n=1}^{\infty} A_n cos(\frac{2\pi nt}{\lambda}) + B_n sin(\frac{2\pi nt}{\lambda}) \tag{9.4.36}$$

Definition 9.4.1. (Periodic Function): A function $f(t)$ is periodic if $\exists\, \lambda \in$ R such that $f(t \pm n\lambda) = f(t)$

14. Cellular Neural Networks (CNN): Being a dynamical system, the CNN can be generally defined using four specifications:

    - Cell dynamics
    - Synaptic law
    - Boundary condition
    - Initial condition

    The internal circuit of the cell is a dynamical system while the dynamics is defined by an evolution equation:

$$\dot{x_i} = -h(x_i, I_i, u_i(t), I_i^s) \tag{9.4.37}$$

15. Typical Methodology in Image Processing: The typical stages used in a computer-based plant species recognition system are shown in Figure 9.2. Going from a physical leaf to knowing its species involves steps such as image acquisition, pre-processing, segmentation, feature extraction and classification.

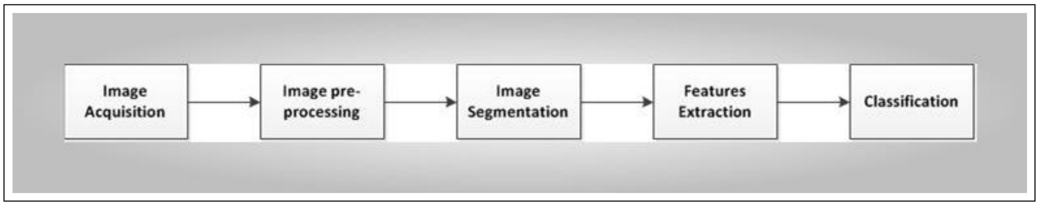


Figure 9.2: Conceptual Diagram for Plant Species Recognition System

16. Other colometric equations : Some other colormetric equations as continued from ?? of chapter 2 are shown in equations 9.4.38, 9.4.38, 9.4.39, 9.4.40, 9.4.41, 9.4.42, & 9.4.43.

    (a) Method 5: Ligthness Method

        The lightness method averages the most prominent and least prominent colors and it's given as

    (b) Method 5

$$Method5 = \frac{(max(R,G,B)+min(R,G,B))}{2} \tag{9.4.38}$$

    (c) Method 6

$$Method6 = max(R,G,B) \tag{9.4.39}$$

    (d) Method 7

$$Method7 = k.\Gamma(max(R,G,B)) \tag{9.4.40}$$

        where $0 < \text{k} \leq 1$

(e) Method 8

$$Method8 = (0.3333R)^a + (0.3333G)^b + (0.3333B)^c \tag{9.4.41}$$

where a, b, and c are all fractions between 0 and 1.

(f) Method 9

$$Method9 = 0.2500R + 0.5000G + 0.2500B \tag{9.4.42}$$

(g) Method 10

$$Method10 = 0.2500R + 0.6250G + 0.1250B \tag{9.4.43}$$

(h) Transformation from RGB to HSV: We assume the triple (r, g, b) define a color in RGB space while (h, s, v) depicts the transformed triple in HSV color space.

$$v = max(r, g, b) \tag{9.4.44}$$

$$s = \frac{v - min(r, g, b)}{v} \tag{9.4.45}$$

$$r' = \frac{v - r}{v - min(r, g, b)} \tag{9.4.46}$$

$$g' = \frac{v - g}{v - min(r, g, b)} \tag{9.4.47}$$

$$b' = \frac{v - b}{v - min(r, g, b)} \tag{9.4.48}$$

$$h = \begin{cases} 5+b' & \text{if} \quad r = max(r,g,b) \quad \text{and} \quad g = min(r,g,b) \\ 1-g' & \text{if} \quad r = max(r,g,b) \quad \text{and} \quad g \neq min(r,g,b) \\ 1+r' & \text{if} \quad g = max(r,g,b) \quad \text{and} \quad b = min(r,g,b) \\ 3-b' & \text{if} \quad g = max(r,g,b) \quad \text{and} \quad b \neq min(r,g,b) \\ 3+g' & \text{if} \quad b = max(r,g,b) \quad \text{and} \quad r = min(r,g,b) \\ 5-r' & \text{otherwise} \end{cases} \tag{9.4.49}$$

Appendix II
A review of some classification models

In this section some other classification models which will be benchmarked against PNN, are briefly described and reviewed.

(a) Naive Bayes Classifier The naive Bayes classifier (NBC) uses Baye's theorem concept to formulate a probabilistic model to estimate the posteriori probability , say, $p(y|x)$ of different $y's$ and predict the one with the largest posterior probability . (This is commonly refered to as the Maximum a Posterior (MAP) rule). The Baye's theorem is simply expressed as:

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)} \tag{9.4.50}$$

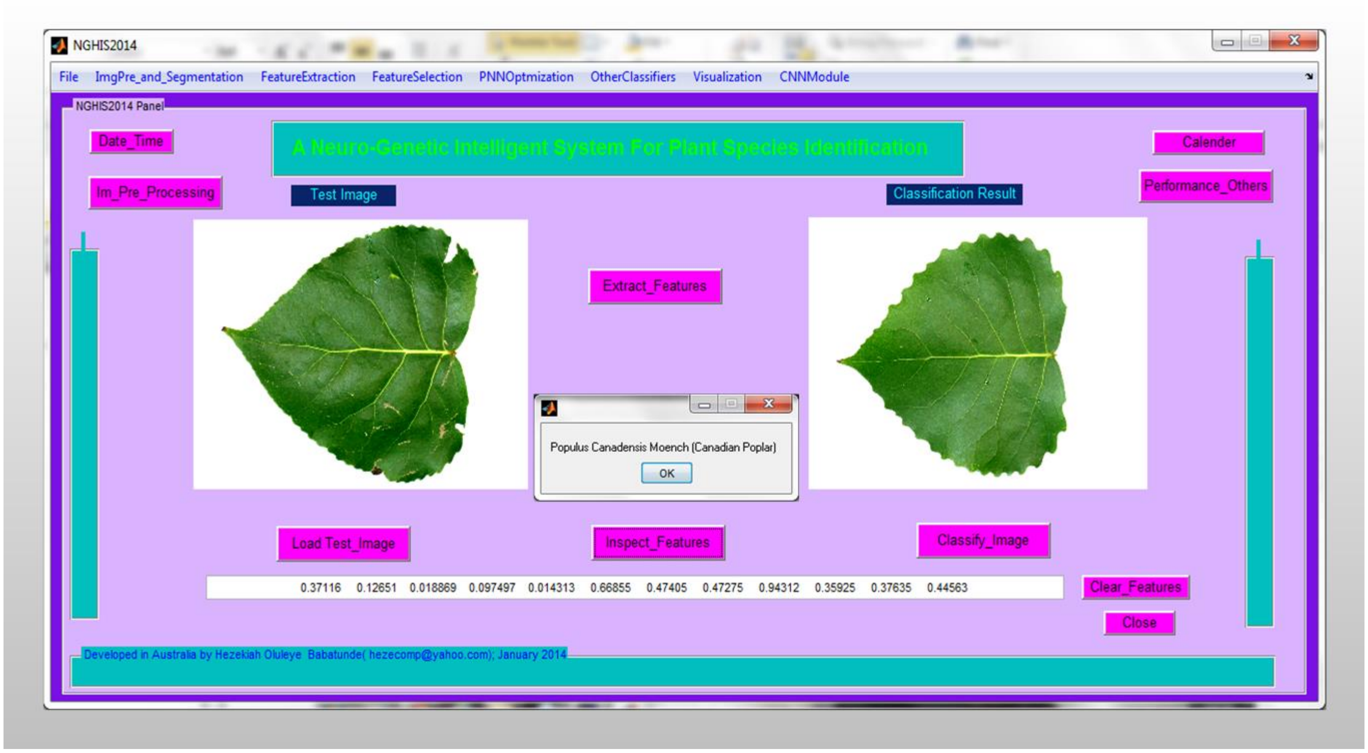
where $p(c)$ can be estimated by counting the proportion of class c in the training set, and $p(x)$ can be ignored since it's a common term across all classes. Unlike the PNN, the NBC assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. With this assumption, conditional distribution over the class variable $c_j, j = 1(1)32$ is given as:

$$p(c_j|X_1, X_2, X_3, ..., X_n) = \frac{1}{\alpha} p(c_j) \prod_{i=1}^{n} p(X_i|c_j) \tag{9.4.51}$$

where $\alpha =$ evidential scale factor depending on the number of features in the training set. Thus the corresponding naive Bayes classifier is given as:

$$classify(x_i \in X) = argmax\ p(c = c_i) \prod_{i=1}^{n} p(X_i = x_i|c = c_i), i = 1(1)n, n > 1\ \&\ n \in Z^+.$$
(9.4.52)

A Naive Bayes classifier assigns a new observation (unseen test data) to the most probable class, assuming the features are conditionally independent given the class value. The NBC herein has three methods viz $f1(.)$, $f2(.)$, & $f3(.)$. The $f1(.)$ fits a NBC to the training data, $f2(.)$ predicts the class label for test data and $f3(.)$ assigns posterior probability for each class of the test data. The posterior probability is defined in section 7.5.13. The prior property for NBC is a vector of length NClasses containing the class priors. In the training phase, the NBC estimates the probabilities $p(c)$ for all classes $c \in C$ and $p(x_i|c)$ for all features $i = 1(1)n$ and all feature values x_i taken from the training set. In the testing phase, an unknown instance x_{test} will be predicted with the label c_{test} if c_{test} leads to the largest value from Equation 9.4.52. An empty class is assigned a prior value of zero. Gaussian distribution was used as the functional for pdf estimation. This classifier is one of the simplest classifiers available but it works incredibly well. NBC was employed in the paper by O. Babatunde et al. (2014d). The screen shot of the image processing tool taken from this paper is given in Figure 9.3.



Populus Canadensis Moench (Canadian Poplar)

0.37116 0.12651 0.018869 0.097497 0.014313 0.66855 0.47405 0.47275 0.94312 0.35925 0.37635 0.44563

Developed in Australia by Hezekiah Oluleye Babatunde(hezecomp@yahoo.com); January 2014

Figure 9.3: Image processing tool on Naive Bayes classifier

(b) Sparse Matrix Representation The sparse matrix approach to image processing does not require all features extracted in chapter 4 of this study, neither does it need any other image descriptors not included herein. It is a kind of feature-less classifier as it works directly on the images via L_1-minimization problem:

$$min||x||_1 \ \ subject\, to \ \ ||y - Ax||_2 \leq \varepsilon \tag{9.4.53}$$

where

i. y = test image

ii. A = matrix representing the image database (or training samples).

iii. x = sparse vector being sought for

Given enough training samples $A \in \mathrm{R}^{M \times N}$ with the associated class information $c_i, i = 1(1)K$, K = number of predictor (class) variables), any given test sample $y \in \mathrm{R}^M$ from the available class will lie in the linear span of the training samples associated with the object, say, i:

$$y = \sum_{i,j}^{M,N} (\alpha_{i,j} V_{i,j}) \tag{9.4.54}$$

for some scalars $\alpha_{i,j}$ and $V_{i,j} \in A$. Since the knowledge of class information for the test image is not available, then a new matrix A for the entire training set is defined and given as the concatenation of all the N training samples of all the k object (image) classes. Thus we have

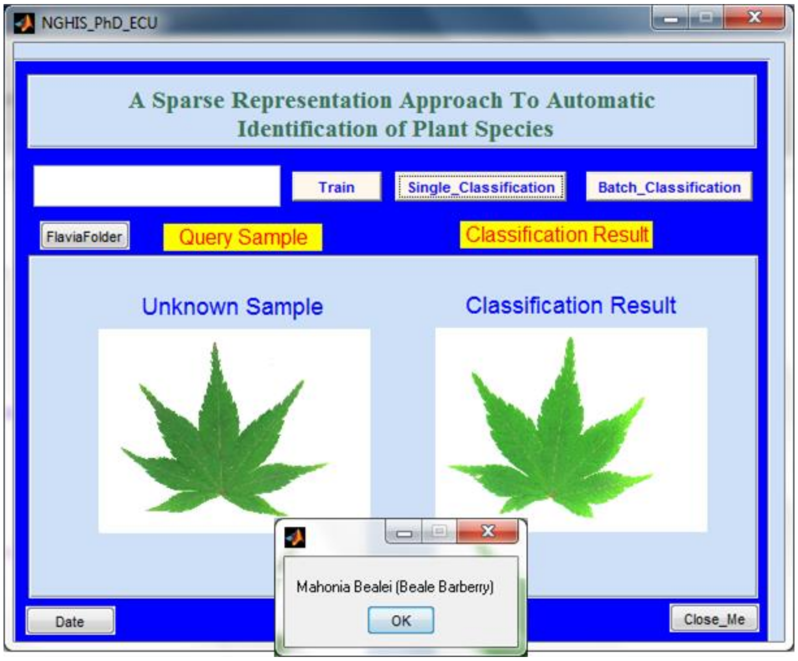
$$A = [A_1, A_2, A_3, ..., A_K] = [V_{1,1}, V_{1,2}, V_{1,3}, ..., V_{K,N}] \tag{9.4.55}$$

Then, then the unknown test sample y can be rewritten in terms of all training samples as:

$$y = Ax_0 \in \mathrm{R}^M \tag{9.4.56}$$

where

$x_0 = [0, ..., 0, \alpha_{i,1}, \alpha_{i,2}, ..., \alpha_{i,N}, 0, ..., 0]^T \in \mathrm{R}^N$ is a cofficient vector whose entries are zero except those associated with the ith class. The screen designed for this classifier is shown below:

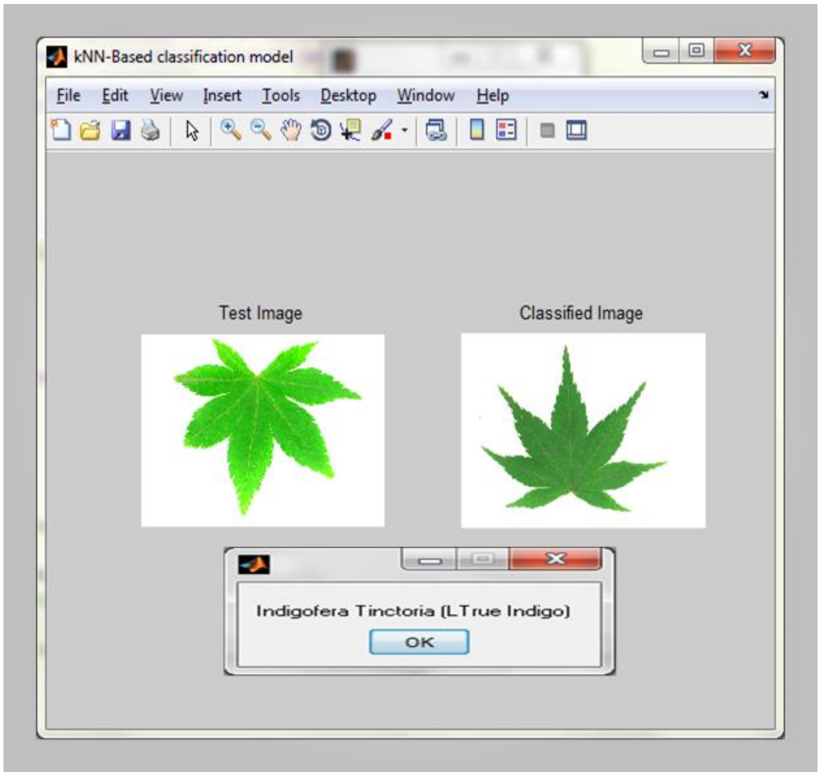


Mahonia Bealei (Beale Barberry)

Figure 9.4: Sparse Matrix Classifier

(c) k Nearest Neighbour (kNN) The kNN is fully described in section 5.4.2 of the Chapter 5 of this thesis. It was used as the fitness function for both GA and PSO-based feature selection. The performance of kNN is also benchmarked against the performance of the main classifier used which is PNN. The different distance metric that can be used with the kNN are Euclidean, standard Euclidean, Mahalanobis, Minkowski, Chebychev, Cosine, Correlation, Jaccard, and Spearman distance. A sample screen shot of the kNN-based classification model used to benchmark the PNN is given in Figure 9.5.

(d) Radial basis network A radial basis function network is a variant of artificial neural network (ANN) that employ radial basis functions as activation functions (Broomhead


kNN-Based classification model
File Edit View Insert Tools Desktop Window Help
Test Image
Classified Image

Indigofera Tinctoria (LTrue Indigo)

Figure 9.5: A kNN-based classfication model

& Lowe, 1988; Mark, 1996; Haykin, 2001). RBF networks are extensively being used in research because they are universal approximator with compact topology. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. The input to RBF could be a feature set or a vector of real numbers $\mathrm{x} \in \mathbb{R}^n$. All the elements of the input vector (s) are algebraically mapped to a scalar output as $\phi : \mathbb{R}^n \to \mathbb{R}$ using the functional in Equation 9.4.57.

$$\phi(\mathrm{x}) = \sum_{i=1}^{K} w_i \rho(||\mathrm{x} - \mathrm{c}_i||) \tag{9.4.57}$$

where K is the number of neurons in the hidden layer, c_i is the center vector for neuron i and w_i is the weight of neuron i in the linear output neuron and ρ is a radius-based metric.. A radial basis function is a real-valued function whose value depends only on the distance from the origin. Thus $\phi(X) = \phi(||X||)$. Functions that depend only on the distance from a center vector are radially symmetric about that vector, hence the name radial basis function. In the basic form all inputs are connected to each hidden neuron. An RBF can be trained without back propagation since it has a closed-form solution. The neurons in the hidden layer contain basis functions. A common basis function for RBF network is a kind of Gaussian function without scaling factor. Further detailes about RBF can be found in (Broomhead & Lowe, 1988; Changbing & Wei, 2010; Kurban & Besdok, 2009; Ackley, 1985; Aleksander, 1989; Amari, 1990). Each node in the input layer of the RBF corresponds to a feature vector from Table 5.1. The second layer is the only hidden layer in the RBF network. The second layer applies non-linear mapping from input vector space into hidden layer space through appropriate non-linear function such as guassian kernel (see equation 9.4.58) which was used in this work. The x in equation 9.4.58 is the training sample, c_i is the hidden ith neuron and σ is the width of the basis function which is a multiple of the average distance between the centers in the RBF network. The σ determines the receptive width of the RBF.

$$h(||x - c_i||) = \exp\left(-\frac{||x - c_i||^2}{2\sigma^2}\right) \tag{9.4.58}$$

$$y(x) = \sum_{i=1}^{K} w_i * h(||x - c_i||) + b \tag{9.4.59}$$

The output layer is made up of neurons that are directly connected to the hidden layer neurons (Demuth et al., 2013; Chen, Cowan, & Grant, 1991). The output value for the training set or any input to the RBF is expressed as equation 9.4.59. The w in equation 9.4.59 is the weight factor normally computed as $w = (h^T h)^{-1} C$ where C is the target class matrix and h^T means ”Transpose of h”. The number of neurons in the output layer is the same as the number of classes in the dataset while the number of neurons in the input layer is equal to the number of features in the training set. Technically speaking, the number of features is equal to the number of features used or selected by optimization techniques such as GA and PSO. For example, GA can be used in determining the σ value while K-means clustering was used in forming the centers in the hidden layer. Typical examples of basis used RBF (taking $d = ||x - x_i||$ & $0 \leq \varepsilon \leq 1$)

are:

i. Gaussian: $\phi(d) = e^{-(\varepsilon d)^2}$

ii. Multiquadratic: $\phi(d) = \sqrt{1+(\varepsilon d)^2}$

iii. Inverse quadratic: $\phi(d) = \frac{1}{1+(\varepsilon d)^2}$

iv. Inverse multiquadratic: $\phi(d) = \frac{1}{\sqrt{1+(\varepsilon d)^2}}$

v. Polyharmonic spline:

A. $\phi(d) = d^n, n = 1, 3, 5, ...$

B. $\phi(d) = d^n \ln(n), n = 2, 4, 6, ...$

vi. Thin plate spline: $\phi(d) = d^2 \ln(d)$

The distance d above can be generally defined as $d(x - x_i) = \left(\sum_{i=1}^{n} |x - x_i|^n\right)^{1/n}$

(e) Multi-layer perceptron A multi-layer perceptron (MLP) is a feedforward ANN consisting of three or more layers of input, output with one or more hidden layers and having non-linear activation nodes. As discussed in chapter 7, a typical feed-forward ANN model (MLP) which has 3 layers viz input, hidden, and output layers with multiple neurons can be represented as

$$y_i = f\left[\sum_{k=1}^{N} \omega_k g\left(\sum_{j=1}^{J} (\omega_j x_j + \phi_j)\right) + \varepsilon_k\right] \tag{9.4.60}$$

where N = Number of hidden-layer neurons, ω_j = synaptic weights connecting the input and hidden layer neurons, ω_k = weights connecting the biases in the hidden and output layers, while $f(.)$ and $g(.)$ are respectively linear and sigmoid functions.

(f) Ensemble Model The rational behind using ensemble model is to combine several classifiers(weak and or strong) into one high-quality ensemble predictor. A supervised learning model is normally equipped with data matrix (dataset), say, X whose rows represent the observations and columns represent features (predictor variable) in the dataset respectively as shown in Figure 9.6. In MATLAB, the ensemble methods that are applicable to classification problems with 2 or more classes are 'AdaBoostM2','LPBoost', 'TotalBoost', 'RUSBoost', 'Subspace', 'Bag'.

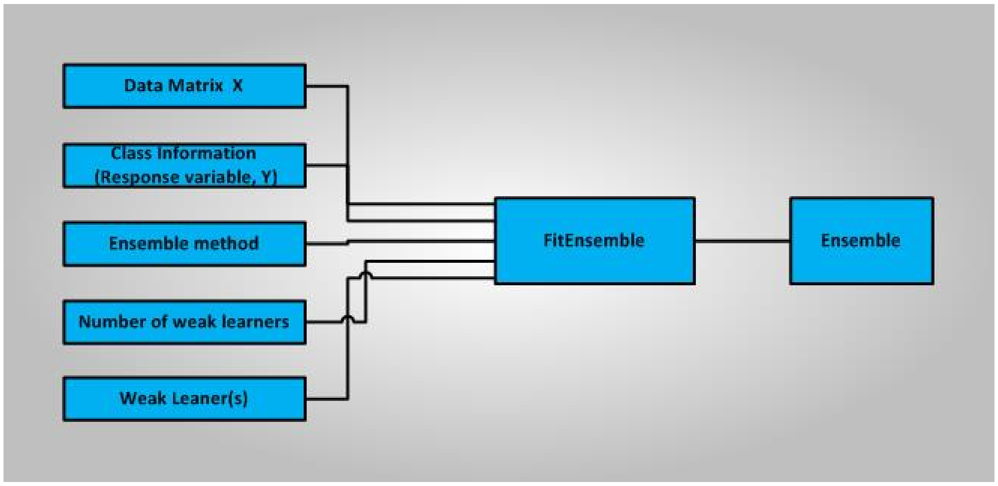


Figure 9.6: Framework for Ensemble Learning

Appendix III
Some code listings
• Conversion from color to grayscale to binary images

```
function BinLeaf = LogicalLeaf3(LeafImage)
if ndims(LeafImage) > 2 % Check for RGB
p11 = rgb2gray(LeafImage); % convert to grayscale
end
if ndims(LeafImage) <=2;
p11 =  LeafImage;
end

level = graythresh(p11);%Otsu Thresholding
LeafBinary = im2bw(p11,level);
originalBW = imcomplement(LeafBinary);%Binary Leaf
%figure, imshow(originalBW);
se = strel(’disk’,10);
BinLeaf   = imclose(originalBW,se);
%imshow(BinLeaf);
end
```

• Genetic Algorithm-Based Feature Selection

```
function Feat_Index = Binary_Genetic_Algorithm_Hezy_2013
% Written by BABATUNDE Oluleye H, PhD Student

% Address: eAgriculture Research Group, School of Computer and Security
% Science, Edith Cowan University, Mt Lawley, 6050, WA, Australia
% Date: 2013
% Please cite any of the article below (if you use the code), thank you

% ”BABATUNDE Oluleye, ARMSTRONG Leisa J, LENG Jinsong and DIEPEVEEN Dean (2014).
% Zernike Moments and Genetic Algorithm: Tutorial and APPLICATION.
% British Journal of Mathematics & Computer Science. 4(15):2217-2236.”

%% OR
%BABATUNDE, Oluleye and ARMSTRONG, Leisa and LENG, Jinsong and DIEPEVEEN (2014).
% A Genetic Algorithm-Based Feature Selection. International Journal of Electronics
% Communication and Computer Engineering: 5(4);889--905.

% DataSet here
%Ionosphere dataset from the UCI machine learning repository:
%http://archive.ics.uci.edu/ml/datasets/Ionosphere
%X is a 351x34 real-valued matrix of predictors. Y is a categorical response:
%”b” for bad radar returns and ”g” for good radar returns.

% NOTE: You can run this code directory on your PC as the dataset is
% available in MATLAB software

clear all
```

```
global Data
% load ionosphere.mat  % This contains X (Features field) and Y (Class Information)
Data  = load('ionosphere.mat'); % This is available in Mathworks
GenomeLength =34; % This is the number of features in the dataset
tournamentSize = 2;
options = gaoptimset('CreationFcn', {@PopFunction},...
'PopulationSize',50,...
'Generations',100,...
'PopulationType', 'bitstring',...
'SelectionFcn',{@selectiontournament,tournamentSize},...
'MutationFcn',{@mutationuniform, 0.1},...
'CrossoverFcn', {@crossoverarithmetic,0.8},...
'EliteCount',2,...
'StallGenLimit',100,...
'PlotFcns',{@gaplotbestf},...
'Display', 'iter');
rand('seed',1)
nVars = 34; %
FitnessFcn = @FitFunc_KNN;
[chromosome,~,~,~,~,~] = ga(FitnessFcn,nVars,options);
Best_chromosome = chromosome; % Best Chromosome
Feat_Index = find(Best_chromosome==1); % Index of Chromosome
end

%%% POPULATION FUNCTION
function [pop] = PopFunction(GenomeLength,~,options)
RD = rand;
pop = (rand(options.PopulationSize, GenomeLength)> RD); % Initial Population
```

```
end

%%% FITNESS FUNCTION   You may design your own fitness function here
function [FitVal] = FitFunc_KNN(pop)
global Data
FeatIndex = find(pop==1); %Feature Index
X1 = Data.X;% Features Set
Y1 = grp2idx(Data.Y);% Class Information
X1 = X1(:,[FeatIndex]);
NumFeat = numel(FeatIndex);
Compute = ClassificationKNN.fit(X1,Y1,’NSMethod’,’exhaustive’,’Distance’,’euclidean’);
Compute.NumNeighbors = 3; % kNN = 3
FitVal = resubLoss(Compute)/(34-NumFeat);
end
```

•Naive Bayes Classifier code

```
function Accuracy = PhD_NaiveBayes
clear all
DataSet = load(’LeafFeatures_Complete_Set.mat’,’-ascii’);
%DataSet = normc(DataSet(:,[1:8]));
%DataSet = load(’LeafColourFeatures.mat’,’-ascii’);
DataSet = DataSet(:,[6    7    8    15    19    55    71    73    74    75    78]);
load(’ClassInfo.mat’) ;
%O1 = NaiveBayes.fit(DataSet,Num);
O1 = NaiveBayes.fit(DataSet,Num, ’dist’,{’kernel’})
C1 = O1.predict(DataSet);
Sample_No = randperm(1907,1)
C2 = O1.predict(DataSet(Sample_No,:))
%POST = posterior(1,DataSet(randperm(1907,1)))
[cMat1,~] = confusionmat(Num,C1); % the confusion matrix
save cMat1
sumi = 0;
for i =1:32
for j = 1:32
if(i==j)
sumi = sumi + cMat1(i,j);
end
end
end
Accuracy = sumi/sum(sum(cMat1));
```

• Comparison of Edge Operators

```
function PhD_Compare_Edge_Operators_March_24_2015
[filename, pathname] = uigetfile({’*.jpg’;’*.*’}, ’File Selector’);
fname = strcat (pathname, filename);
Im = imread(fname);
%imshow(Im1);
Im = rgb2gray(Im);
Im = LogicalLeaf3(Im);

Sobel_Edge = edge(Im,’sobel’);
Canny_Edge = edge (Im, ’canny’);
Prewitt_Edge = edge(Im,’prewitt’);
LoG_Edge = edge(Im,’log’);

tic;
tstart1 = tic;
edge(Im,’sobel’);
telapsed_sobel = toc(tstart1);

tic;
tstart2 = tic;
edge(Im,’canny’);
telapsed_canny = toc(tstart2);

tic;
tstart3 = tic;
edge(Im,’prewitt’);
telapsed_prewitt = toc(tstart3);
```

```
tic;
tstart4 = tic;
edge(Im,’log’);
telapsed_LoG = toc(tstart4);


subplot (2,2,1);
imshow(Sobel_Edge);
title (’Sobel edge output’);
xlabel([’Computational time in secs: ’, num2str(telapsed_sobel)]);

subplot(2,2,2);
imshow (Canny_Edge);
title(’Canny edge output’);
xlabel([’Computational time secs: ’ , num2str(telapsed_canny)]);

subplot(2,2,3);
imshow (Prewitt_Edge);
title (’Prewitt edge output’);
xlabel([’Computational time in secs: ’, num2str(telapsed_prewitt)]);
subplot(2,2,4);
imshow(LoG_Edge);
title(’LoG edge output’);
xlabel([’Computational time in secs: ’ , num2str(telapsed_LoG)]);
```

•Code for Fourier Descriptors

```
function [New_fd]  = PhD_FD_Features(Im1)


img = Im1;
if ndims(img)>2
img = NormalLeaf(img);
end
I = img;
n=size(img) ;
b=I;
% edge detection
a=double(I);
for x=2:n(1)-1
for y=2:n(2)-1
c=abs(a(x+1,y+1) - a(x,y))+ abs(a(x+1,y) - a(x,y+1)) ;
b(x,y)=c;
end
end



nf=21; % no of boundary points
[rows, cols] = find(I~=0);
contour = bwtraceboundary(I, [rows(1), cols(1)], ’N’,8,nnz(I),’clockwise’);
sampleFactor = round(length(contour)/nf);


dist = 1;
% counting the FDs
for i=1:nf
```

```
c(i) = contour(round(dist),2) + 1i*contour(round(dist),1);
dist = dist + sampleFactor;
%end
end
C = fft(c);
Capprox = C;
LeafFDs = abs(Capprox);
LeafFDs = LeafFDs(1:nf);
LeafFDs(1) = 0;
LeafFDs = LeafFDs/(LeafFDs(2));
New_fd  = LeafFDs;
disp(New_fd);

end
```

- Image Moment

```
function LeafMoment = LeafRawMoment(LeafBinary,i,j)

LeafMoment = sum(sum( ((1:size(LeafBinary,1))'.^j * (1:size(LeafBinary,2)).^i) .* LeafBinary ));

end

function LeafCentralMoment = central_moments(LeafBinary,i,j)

LeafRM00 = LeafRawMoment(LeafBinary,0,0);

LeafRM01 = LeafRawMoment(LeafBinary,0,1);

LeafRM10 = LeafRawMoment(LeafBinary,1,0);

centroids = [LeafRM10/LeafRM00 , LeafRM01/LeafRM00];

LeafCentralMoment = sum(sum( (([1:size(LeafBinary,1)]-centroids(2))'.^j * ...

([1:size(LeafBinary,2)]-centroids(1)).^i) .* LeafBinary ));

end


% % % Function for Central Moment


function LeafCentralMoment = central_moments(LeafBinary,i,j)

LeafRM00 = LeafRawMoment(LeafBinary,0,0);

centroids = [LeafRawMoment(LeafBinary,1,0)/LeafRM00 , LeafRawMoment(LeafBinary,0,1)/LeafRM00

LeafCentralMoment = sum(sum( (([1:size(LeafBinary,1)]-centroids(2))'.^j * ...

([1:size(LeafBinary,2)]-centroids(1)).^i) .* LeafBinary ));

end
```

.Code for login menu GUI

```
function varargout = NGHIS2015(varargin)
% NGHIS2015 MATLAB code for NGHIS2015.fig
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct(’gui_Name’,       mfilename, ...
’gui_Singleton’,  gui_Singleton, ...
’gui_OpeningFcn’, @NGHIS2015_OpeningFcn, ...
’gui_OutputFcn’,  @NGHIS2015_OutputFcn, ...
’gui_LayoutFcn’,  [] , ...
’gui_Callback’,   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before NGHIS2015 is made visible.
function NGHIS2015_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for NGHIS2015
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes NGHIS2015 wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = NGHIS2015_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in Login_Button.
function Login_Button_Callback(hObject, eventdata, handles)
msgbox('Stage 1 Login')
pass = passcode;
% %H = NGHISPhDThesisECU
if strcmp(pass,'Hezekiah')
msgbox('Stage 2 Login')
NGHIS2014
end
if ~strcmp(pass,'Hezekiah')
close
errordlg('You cannot continue now; please login again')
handles.output = hObject;
guidata(hObject, handles);
end
```

```
% --- Executes on button press in Close_Button.
function Close_Button_Callback(hObject, eventdata, handles)
close
```

References

Abdul, K. (2014). A model of plant identification system using glcm, lacunarity and shen features. Research Journal of Pharmaceutical, Biological and Chemical Sciences., 5(2), 1-10.

Abdul, K., Lukito, E. N., Adhi, S., & Santosa, P. I. (2012). Experiments of zernike moments for leaf identification. Journal of Theoretical and Applied Information Technology, 41(1), 83-93.

Abramowitz, M., & Stegun, I. A. (1965). Handbook of mathematical functions. Dover Publications.

Ackley, D. H. (1985). A learning algorithm for boltzmann machines. Cognitive Science, 9, 147-169.

Aha, D. (1997). Lazy learning. Kluwer Academic Publishers, Dordrecht.

Aizenberg, I. (2001). Advances in neural networks. Temples Project JEP-16160-2001, 1-58.

Aldea, R., Fira, M., & Lazar, A. (2014, Nov). Classifications of motor imagery tasks using k-nearest neighbors. In Neural network applications in electrical engineering (neurel), 2014 12th symposium on (p. 115-120). doi: 10.1109/NEUREL.2014.7011475

Aleksander, I. (1989). Neural computing aarchitecture: The design of brain-like machines. (Aleksander, Ed.). North Oxford, London.

Aliaga, D. G. (2010). Color and perception. CS635 Spring 2010, Department of Computer Science Purdue University, 1-34.

Alireza, F., Jean, C. C., & Kyandoghere, K. (2011). Cellular neural network trainer and template optimization for advanced robot locomotion, based on genetic algorithm. Transportation Informatics Group, University of Klagenfurt-Austria, 1-6.

Amari, S. I. (1990). Mathematical foundations of neurocomputing. Proc.IEEE, 78, 1443-1463.

Amran, A. H., & Prema, L. S. (2008). Moving object detection using cellular neural networks (cnn). Faculty of Electrical & Electronics Engineering ,University Malaysia Pahang.

Anagnostopoulos, C., Anagnostopoulos, I., Loumos, V., & Kayafas, E. (2006, Sept). A license plate-recognition algorithm for intelligent transportation system applications. Intelligent Transportation Systems, IEEE Transactions on, 7(3), 377-392. doi: 10.1109/TITS.2006 .880641

Andreas, B., Asuka, K., Marion, B., Nick, M., Guido, S., & Andrew, F. (2010). Leafprocessor: a new leaf phenotyping tool contour bending energy and shape cluster analysis. New Phytologist, 187, 251-261.

Arora, A., Gupta, A., Bagmar, N., Mishra, S., & Bhattacharya, A. (2012). A plant identication system using shape and morphological features on segmented leaves:team iitk, clef 2012. Department of Computer Science and Engineering,Indian Institute of Technology, Kanpur, India and Department of Computer Science and Engineering,University of Florida, Gainesville, USA, 1-14.

Awad, K., Chehdi. (2009). Satellite image segmentation using variable hybrid genetic algorithm. Wiley International Journal of Imaging Systems and Technology., 19, 199-207.

Babatunde, H. O., Akanbi, C. O., Fadare, O. G., Eludire, A. A., Aluko, O. B., & Egbedokun, O. G. (2012). On numerical simulation of a boundary-valued neuronal model. World J of Engineering and Pure and Applied Sci, WJEPAS, 20(2), 20-25.

Babatunde, O., Armstrong, L., Diepeveen, D., & Leng, J. (2015). A neuronal classification system for plant leaves using genetic image segmentation. British Journal of Mathematics & Computer Science, 9(3), 261-278. doi: 10.9734/BJMCS/2015/14611

Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014a). Application of cellular neural networks and naivebayes classifier in agriculture. In proceedings of AFITA 2014, 9th Conference of the Asian Federation for Information Technology in Agriculture, Australia, Perth, 6 - 9 October 2014, 63-72.

Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014b). A genetic algorithm-based feature selection. International Journal of Electronics Communication and Computer Engineering, 5, 889–905.

Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014c). On the application of genetic probabilistic neural networks and cellular neural networks in precision agriculture. Asian Journal of Computer and Information Systems, 2(4), 90-100.

Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2014d). Zernike moments and genetic algorithm: Tutorial and application. British Journal of Mathematics and Computer Science, 4(15), 2217-2236.

Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2015a, February). Comparative analysis of genetic algorithm and particle swam optimization: An application in precision agriculture. Asian Journal of Computer and Information Systems, 3(1), 1-12.

Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2015b). A computer-based vision systems for automatic identification of plant species using knn and genetic pca. Journal of Agricultural Informatics, 6(2), 32-44.

Babatunde, O., Armstrong, L., Leng, J., & Diepeveen, D. (2015c). A survey of computer-based vision systems for automatic identification of plant species. Journal of Agricultural Informatics, 6(1), 61-71. doi: 10.17700/jai.2015.6.1.152

Balya, D., & Roska, T. (1999). Face and eye detection by cnn algorithms. J. VLSI Signal Process. Syst. 23, 2(3), 497-511.

Bao, S., Forrest, Lie, D. C., & Zhang, Y. (2008). A new approach to automated epileptic diagnosis using eeg and probabilistic neural network. In Tools with Artificial Intelligence. ICTAI'08. 20th IEEE International Conference, 2, 482-486.

Basili, V. R. (1993). The experimental paradigm in software engineering. Conference Proceedings of Dagstuhl-Workshop: Experimental Software Engineering Issues: Critical Assessment and Future Directives, 1-7.

Belhumeur, P. N., Daozheng, C., Steven, F., David, W., Jacobs, W., John, K., ... Ling, Z. (2008). Searching the world's herbaria: A system for visual identification of plant species. In Computer VisionCV 2008, Springer Berlin Heidelberg., 116-129.

Belhumeur, P. N., & David, J. W. (2011). An electronic field guide: Plant exploration and discovery in the 21st century) and by the washington biologists' field club. http://leafsnap.com/; NSF

Grant IIS-03-25867. (Scientists at Columbia University, the University of Maryland, and the Smithsonian Institution)

Best, D., & Roberts, D. (1975). Algorithm as 89: The upper tail probabilities of spearman's rho". Applied Statistics(24), 377-379.

Biey, M., Checco, P., & Gilli, M., M. (2003). Bifurcations and chaos in cellular neural networks. Journal of Circuits, Systems and Computers, 12(04), 417-433.

Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford university press.

Brendo, J. W., Nikola, K. K., & Howard, C. (2011). Fruit image analysis using wavelets. Wearing.

Broomhead, D. S., & Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Royal Signals and Establishment Malvern (United Kingdom), No. RSRE-MEMO-4148.

Bruzzone, L., & Persello, C. (2010). A novel approach to the selection of robust and invariant features for classification of hyperspectral images. Department of Information Engineering and Computer Science, University of Trento.

Bucolo, M., Caponetto, R., Fortuna, L., & Frasca, M. (2005). The cnn paradigm: Shapes and complexity. International Journal of Bifurcation and Chaos, 15(7), 2063-2090.

Cambell, & Atchlev. (1981). Geometry of principal component. Systematic Zoology, 30(3), 268-280.

Canny, J. (1986). A computational approach to edge detection. IEEE Trans. PAMI, 8(6), 679-698.

Cerutti, G., Tougne, L., Mille, J., Vacavant, A., & Coquin, D. (2013). A model-based approach for compound leaves understanding and identification. In International Conference on Image Processing (ICIP)., 1471-1475.

Changbing, L., & Wei, H. (2010). Application of genetic algorithm-rbf neural network in water environment risk prediction. In Computer Engineering and Technology (ICCET). 2nd International Conference on., 7, 239-242.

Charles, T. Z., & Ralph, Z. R. (1972). Fourier descriptors for plane closed curves. IEEE Transactions on Computers, C-21(3), 269-281.

Charters, J., Wang, Z., Chi, Z., Tsoi, A. C., & Feng, D. D. (2014). Eagle: A novel descriptor for identifying plant species using leaf lamina vascular features. In In multimedia and expo workshops (icmew), 2014 ieee international conference on (pp. 1-6). ieee.

Chen, S., Cowan, C., & Grant, P. (1991). Orthogonal least squares learning algorithm for radial basis function networks. IEEE Transactions on Neural Networks., 2(2), 302-309.

Cheung, V., & Cannons, K. (2002). An introduction to probabilistic neural networks. Retrieved August, 2.

Chomtip, P., Chawin, K., Pitchayuk, S., & Nititat, S. (2011). Leaf and flower recognition system (e-botanist). IACSIT International Journal of Engineering and Technology, 3(4), 10-15.

Chomtip, P., Supolgaj, R., Piyawan, T., & Chutpong, C. (2011). Thai herb leaf image recognition system(thlirs). Kasetsart J. (Nat. Sci.), 45, 551 - 562.

Chong, C. W., Raveendran, P., & Mukundan, R. (2004). Translation and scale invariants of legendre moments. Pattern Recognition., 37(1), 119-129.

Christopher, K., & Garrison, W. (2012). Color-to-grayscale: Does the method matter in image recognition. Plos One, 7(1).

Chua, L. O., & Roska, T. (2002). Cellular neural networks and vision computing. Cambridge University Press.

Chua, L. O., & Yang, L. (1988). Cellular neural networks: theory and applications. IEEE Trans on Circuits and System, 35(10), 1257-1272.

Clarke, B., Fokoue, E., & Zhang, H. (2009). Principles and theory for data mining and machine learning. Springer Series in Statistics, http://www.amazon.com/Principles-Machine-Learning-Springer-Statistics/dp/0387981349, Page 798.

Cope, J. S., Corney, D., Clark, J. Y., & Remagnino, P. W. (2012). Plant species identification using digital morphometrics: A review. Expert Systems with Applications, 7562-7573.

Cordon, O., Herrera, DelJesus, M. J., & Villar, P. (2001). A multi-objective genetic algorithm for feature selection and granularity learning in fuzzy-rule based classication system. IEEE, 1253-1258.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. Information Theory, IEEE

Transactions on, 13(1), 21-27.

De Mantras, R., & Armengol, E. (1998). Machine learning from examples: inductive and lazy methods. Data knowl Eng, 25, 99-123.

Demuth, H., Beale, M., & Hagan, M. (2013). Neural network toolbox userguide. The MathWorks. Inc., Natick, MA.

Dengsheng, Z., & Guojun, L. (2000). A comparative study on shape retrieval using fourier descriptors with different shape signatures. Gippsland School of Computing and Information Technology,Monash University,Australia.

Dobrescu, R., Dobrescu, M., Mocanu, S., & Popescu, D. (2010). Medical images classification for skin cancer diagnosis based on combined texture and fractal analysis. WISEAS Transactions on Biology and Biomedicine, 7(3), 223-232.

Du, J. (2007). Leaf shape based plant species recognition. Applied Mathematics and Computation, Elsevier, 185(2), 883-893.

Duraisamy, M., & Duraisamy, S. (2012). Cnn-based approach for segmentation of brain and lung mri images. European Journal of Scientific Research, 81(3), 298-313.

Dutta, R., Hines, E. L., Gardner, J. W., & Boilot, P. (2002). Bacteria classification using cyranose 320 electronic nose. Biomedical engineering online., 1(4), 1-7.

Eiben, A. E., & Smith, J. E. (2010). Introduction to evolutionary computing (G. Rozenberg, Ed.). Springer-Verlag Berlin Heidelberg.

Ercsey, R., Maria, M., Néda, Z., & Roska, T. (2008). Statistical physics on cellular neural network computers. Physica D: Nonlinear Phenomena, 237(9), 2051-2068.

Fan, J., Peng, J., Gao, L., & Zhou, N. (2015). Hierarchical learning of tree classifiers for large-scale plant species identification. In In semantic computing (icsc), 2015 ieee international conference (p. 389-396).

Flicker, M., Sawhney, H., & Niblack, W. (1996). Query by image and video content: the qbic system. IEEE Computer, 23(9), 23-32.

Flusser, J. (2000). On the independence of rotation moment invariants. Pattern Recognition, 33, 1405-1410.

Flusser, J., Suk, T., & Zitova, B. (2009). Moments and moment invariants in pattern recognition. A John Wiley and Sons, Ltd, Publication, 1-303.

Fourier, J. B. (1878). The analytical theory of heat. The University Press.

Garcia, J., & Barbedo, A. (2013). Digital image processing techniques for detecting, quantifying and classifying plant ddisease. SpringerPlus, 2, 660.

Gebhardt, S. J. L. R. . K. W., Steffen. (2006). Identification of broad-leaved dock (rumex obtusifolius l.) on grassland by means of digital image processing. Precision Agriculture, 7, 165-178.

George, E. (2011). Machine vision identification of plants; recents trends for enhancing the diversity and quality of soyabean products. (P. Krezhova, Ed.). InTech, Available from http://www.intechopen.com/books/recent-trends-for-enhancing-the-diversity-and-quality-of-soyabean-p

Georgiou, V. L., Malefaki, S. N., Alevizos, P., & Vrahatis, M. N. (2006). Evolutionary bayesian probabilistic neural networks. ICNAAM, 393-396.

Gibbons, J. D., & Chakraborti, S. (2011). Nonparametric statistical inference. Springer Berlin Heidelberg (pp. 977-979).

Gilli, M., Roska, T., Chua, Y., Leon, O., Civalleri, B., & Pier, P. (2002). Cnn dynamics represents a broader class than pdes. International Journal of Bifurcation and Chaos, 12(10), 2051-2068.

Gish, H. (1990). A probabilistic approach to the understanding and training of neural network classifiers. IEEE International Conference In Acoustics, Speech, and Signal Processing. ICASSP-90., 4, 1361-1364.

Goeau, H., Bonnet, P., Joly, A., Boujemaa, N., Barthelemy, D., Molino, J.-F., … Picard, M. (2011). The imageclef 2011 plant images classication task. ImageCLEF 2011, 1-19.

Gonzalez. (2007). Mathematical morphology. Chapter 9 of Digital Image processing, INF3300 /INF4300, 1-12.

Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2009). Digital image processing using matlab. Gatesmark Publishing, A Division of Gatesmark LLC, USA.

Gorunescu, F. (2006). Benchmarking probabilistic neural network algorithms. In Proceedings of

International Conference on Artificial Intelligence and Digital Communications., 1-7.

Goto, T., Hirano, S., & Sakurai, M. (2014, December). Face image processing by tv filter and super-resolution. In Visual Communications and Image Processing Conference, IEEE, 245-248.

Govaerts, R. (2001). How many species of seed plants are there? Taxon, 50, 1085-1090.

Gromski, P. S., Xu, Y., Correa, E., Ellis, D. I., Turner, M. L., & Goodacre, R. (2014). A comparative investigation of modern feature selection and classification approaches for the analysis of mass spectrometry data. Analytica chimica acta, 829, 1-8.

Grother, P. J., Candela, G. T., & Blue, J. L. (1997). Fast implementations of nearest neighbor classifiers. Pattern Recognition., 30(3), 459-465.

Gu, W. (2005). Leaf recognition based on the combination of wavelet transform and gaussian interpolation. ICIS, 36(4), 253-262.

Guo, W. W. (2010). Incorporating statistical and neural network approaches for student course satisfaction analysis and prediction. Expert Systems with Applications, 37(4), 3358-3365.

Han, L., Embrechts, M. J., & Szymanski, B. (2011). Sigma tuning of gaussian kernels: Detection of ischemia from magnetocardiograms. Computational Modeling and Simulation of Intellect: Current State and Future Perspectives IGI Global, 206-223.

Hanggi, M., & Moschytz, G. S. (1997). Visualisation of cnn dynamics. Electronics Letters, 33(20).

Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image classification. IEEE T Syst Man Cyb, 3(6), 610-621.

Haykin, S. (2001). Redes neurais - princos e prcas. Bookman.

He, Y., Fataliyev, K., & Wang, L. (2013, January). Feature selection for stock market analysis. In Neural Information Processing. Springer Berlin Heidelberg., 737-744.

Hezekiah, B., Akinwale, A. T., & Folorunso, O. (2010). A cellular neural networks- based model for edge detection. Journal of Information and Computing Science, 5(1), 003-010.

Holland, J. (1962). Genetic algorithms. Scientific American.

Hollander, M., Wolfe, D. A., & Chicken, E. (2013). Nonparametric statistical methods. John Wiley & Sons.

Honfei. (2010). Classification of camellia(theaceae)species using leaf architecture variations and pattern recognition. IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), 2(2), 332-335.

Hosny, K. M. (2007). Exact legendre moment computation for gray level images. Pattern Recognition., 40(12), 3597-3605.

Hu, M. K. (1962). Visual pattern recognition by moment invariants. IEEE Transactions on Information Theory, 8(1), 1-13.

Huang, C. J. (2002). A performance analysis of cancer classification using feature extraction and probabilistic neural networks. In Proceedings of the 7th Conference on Artificial Intelligence and Applications., 374-378.

Jiang, Y., Wang, R., & Zhang, P. (2008). Texture description based on multiresolution moments of image histograms. Optical Engineering, 47(3), 037005-037005-7. Retrieved from http://dx.doi.org/10.1117/1.2894149 doi: 10.1117/1.2894149

John, W., Allen, Y., Arvind, G., Shankar, S., & Yi, M. (2009). Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 31(2), 1-17.

Jyotismita, C., & Ranjan, P. (2011). Plant leaf recognition using shape based features and neural network classifiers. International Journal of Advanced Computer Science and Applications (IJACSA), 41-47.

Kadir, A. (2011). Neural network application on foliage plant identification. , Indonesia.

Kadir, A. (2015). Leaf identification using fourier descriptors and other shape features. Gate to Computer Vision and Pattern Recognition (gtCVPR)., 1(1), 3-7. doi: doi:10.15579/gtcvpr.0101.003007

Kawulok, M., & Nalepa, J. (2014a). Hand pose estimation using support vector machines with evolutionary training. In Systems, Signals and Image Processing (IWSSIP), 2014 International Conference on IEEE., 87-90.

Kawulok, M., & Nalepa, J. (2014b). Hand pose estimation using support vector machines with evolutionary training. In In systems, signals and image processing (iwssip), 2014 international conference on (pp. 87-90). ieee.

Kekre, H. B., Thepade, T. K., & Sarode, V. S. (2010). Image retrieval using texture features extracted from glcm,lbg, and kpe. International Journal of Computer Theory and Engineering, 2(5), 695-700.

Kendall, M. (1970). Rank correlation methods. Griffin.

Kenji, O., & Morio, O. (1984). Measurement of stomatal aperture by digital image processing. Plant & Cell Physiol.JSPP, 25(8), 1379-1388.

Kim, D. K., Lee, J. J., Lee, J. H., & Chang, S. K. (2005). Application of probabilistic neural networks for prediction of concrete strength. Journal of Materials in Civil Engineering; ASCE, 17, 353-362.

Kittler, J. (1978). Feature set search algorithms. Pattern Recognition and Signal Processing. Sijhoff an Noordhoff, the Netherlands.

Kohavi, R., & John, G. (1996). Wrappers for feature subset selection. . Artificial Intelligence, special issue on relevance, 97(1-2), 273-324.

Kohavi, R., & John, H. G. (1997). Wrappers for feature subset selection. Elsevier Artificial Intelligence, 97, 273-324.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. Informatica, 31, 249-268.

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, 26(3), 159-190.

Kpalma, K., & Ronsin, J. (2007). An overview of advances of pattern recognition systems in computer vision. Vision Systems: Segmentation and Pattern Recognition, ISBN 987-3-902613-05-9., 169-194.

Krose, B., & van der Smagt, P. (1996). An introduction to neural networks. [Online]. Available: citeseer. ist.psu.edu/article/krose93introduction.html, 1-135.

Kulkarni, A., Rai, H. M., Jahagirdar, K. A., & Upparamani, P. S. (2014). A leaf recognition technique for plant classification using rbpnn and zernike moments. International Journal of Advanced Research in Computer and Communication Engineering, 2(1), 984-988.

Kumar, N., Belhumeur, P. N., Biswas, A., Jacobs, D. W., Kress, W. J., Lopez, I., & Soares, V. B. (2011). Leafsnap: A computer vision system for automatic plant species identication. University of Washington, Seattle WA, Columbia University, New York NY, University of Maryland, College Park MD, National Museum of Natural History, Smithsonian Institution, Washington DC, 1-14.

Kurban, T., & Besdok, E. (2009). A comparison of rbf neural network training algorithms for inertial sensor based terrain classification. Sensors, 6312-6329.

Laga, H., Kurtek, S., Srivastava, A., Golzarian, M., & Miklavcic, S. (2012). A riemannian elastic metric for shape-based plant leaf classification. DICTA 2012 Conference Proceedings, 2012, 1-7.

Lal, T. N., Chapelle, O., Weston, J., & Elisseeff, A. (2006). Embedded methods. In Feature extraction. Springer Berlin Heidelberg., 137-165.

Leng, J., Valli, C., & Armstrong, L. (2010). A wrapper-based feature selection for analysis of large data sets. In 2010 3rd international conference on computer and electrical engineering (iccee 2010) (p. 166-170).

Ling, H., & Jacobs, D. W. (2007). Shape classification using the inner-distance. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 29(2), 286-299..

Luigi, F., Paolo, A., David, B., & Akos, Z. (2001). Cellular neural networks : A paradigm for nonlinear spatio-temporal processing. IEEE Proceedings.

Luminita, V. (2010). An introduction to mathematical image processing ias (Tech. Rep.). Park City Mathematics Institute, Utah Undergraduate Summer School.

Maldonado, S., & Weber, R. (2009). A wrapper method for feature selection using support vector machines. Information Sciences, 179(13), 220817.

Maldonado, S., & Weber, R. (2011). Embedded feature selection for support vector machines: state-of-the-art and future challenges. In In progress in pattern recognition, image analysis, computer vision, and applications. (p. 304-311). Springer Berlin Heidelberg.

Maleki, A., & Do, T. (2009). Review of probability theory. Computing in Science & Engineering, 11(1), 8-18.

Mallah, C., Cope, J., & Orwell, J. (2013). Plant leaf classification using probabilistic integration of shape, texture and margin features. Signal Processing, Pattern Recognition and Applications.

Marek, O. (1998). Introduction to genetic algorithms. Czech Technical University (http://www.obitko.com/tutorials/genetic-algorithms/about.php).

Mariofanna, M., Paulo, E. M., Almeida, Jun, O. J., & Marcelo, G. S. (1999). Applications of cellular neural networks for shape from shading problem.

Mark, J. L. (1996). Introduction to radial basis function networks. Recent Advances in Radial Basis Function Networks. Centre for Cognitive Science, University of Edinburgh, Scotland, 1-67.

Martinez, W., & Martinez., A. (2002). Computational statistics handbook with matlab. CRC Press, Bocca Raton.

Mathsworks. (2013). Statistics toolbox: User's guide for version r2013a. The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098.

MathWorks. (2007). Matlab neural network toolbox documentation. MathWorks. Inc. [Online]. Available:.

Mathworks. (2009). Signal processing toobox (discrete fourier transform (dft)). MathWorks. Inc.

MathWorks. (2013). Genetic algorithm. Global Optimization Toobox.

Mathworks. (2015). Matlab : The language of technical computing. http://au.mathworks.com/.

McBratney, A., Whelan, B., & Ancev, T. (2005). Future directions of precision agriculture. Precision Agriculture., 6, 7-23.

Meeta, K., Mrunali, K., Shubhada, P., Prajakta, P., & Neha, B. (2012). Survey on techniques for plant leaf classification. International Journal of Modern Engineering Research (IJMER), 1(2), 538-544.

Melanie, M. (1999). An introduction to genetic algorithms. A Bradford Book The MIT Press.

Navidi, W. C. (2015). Statistics for engineers and scientists. McGraw-Hill Higher Education.

Niculescu, S. P., Lewis, M. A., & Tigner, J. (2008). Probabilistic neural networks modelling of the 48-h lc50 acute toxicity endpoint to daphnia magna. SAR, 19(7-8), 735-750.

Nixon, M. S., & Aguado, A. S. (2002). Feature extraction and image processing. Newnes.

Nixon, M. S., & Aguado, A. S. (2012). Feature extraction & image processing for computer vision. Elseviet Ltd, the Boulevard, Langford Lane, Kindlington, Oxford, OX5 1GB , UK.

Noll, R. J. (1976). Zernike polynomials and atmospheric turbulence. JOsA, 66(3), 207-211.

Onkar, N. R. (2011). Shape recognition for plane closed curves using error model of an elliptical fit and fourier descriptors (Unpublished master's thesis). Master of Science Thesis in the department Electrical Engineering,University of North Carolina at Charlotte.

Orwell, J., Mallah, C., & Cope, J. (2012). Plant leaf classification using probabilistic integration of shape, texture and margin features. Signal Processing, Pattern Recognition and Applications, in press..

Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE Trans. Sys., Man., Cyber; doi:10.1109/TSMC.1979.4310076., 9(1), 62.

Pahalawatta, K. K. (2008). A plant identification system using both global and local features of plant leaves. MSc Thesis at the department of Computer Science and Software Engineering, University of Canterbury, New Zealand, 1-127.

Panagiotis, T. (2005). Plant leaves classification based on morphological features and a fuzzy surface selection techniques. Fifth International Conference on Pattern Recognition, Greece, 365-370.

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization method for constrained optimization problems. Intelligent Technologieseory and Application: New Trends in Intelligent Technologies., 76, 214-220.

Parzen, E. (1962). On estimation of a probability density function and mode. Annals of Mathematical Statistics, Vol 33, Issue 3, 1065-1076.

Pat, G. (2000). Plant identification: Examiming leaves. Oregon State Univesity, Department of Horticulture.

Peters, R. A. (2011). Lecture notes: The 1 and 2 - dimensional fourier transforms. EECE-CS 253 Image Processing, Vanderbilt University School of Engineering, 1-95.

Ping Tian, D. (2013). A review on image feature extraction and representation techniques.

International Journal of Multimedia and Ubiquitous Engineering, 8(4), 385-396.

Pinkus, A. (1999). Approximation theory of the mlp model in neural networks. Acta Numerica, 143-196.

Pookhao, N., Sohn, M., Li, J. I., Q, Du, R., Jiang, H., & An, L. (2015, Jan 15). A two-stage statistical procedure for feature selection and comparison in functional analysis of metagenomes. Bioinformatics, 31(2), 158-165. doi: doi:10.1093/bioinformatics/btu635 .Epub2014Sep24.

Prasad, S., Peddoju, S. K., & Ghosh, D. (2013). Mobile plant species classification: A low computational aproach. In Image Information Processing (ICIIP), 2013 IEEE Second International Conference., 405-409.

Prewitt, J. M. S., & Mendelsohn, M. L. (1966). Analysis of cell images. Ann. N.Y Acad. Sci., 128, 1035-1053.

Pundkar, S. V., & Waghmare., M. M. (2014). Study of various techniques for medicinal plant identification. International Journal on Recent and Innovation Trends in Computing and Communication., 2(11), 3340 - 3343.

Quadri, A. T., & Sirshar, M. (2015). Leaf recognition system using multi-class kernel support vector machine. International Journal of Computer and Communication System Engineering., 2(2), 260-263.

Rashad, M. Z., El-Desouky, B. S., & Khawasik, M. S. (2011). Plants images classification based on textural features using combined classifier. International Journal of Computer Science & Information Technology (IJCSIT), 3(4), 93-100.

Richard, E. B., & Stuart, E. D. (1962). Applied dynamic programming (Tech. Rep.). United State Airforce Project, Rand Corporation.

Roska, T., Zarandy, A., & Rekeczky, C. (2003). Cellular neural networks. CRC Press LLC.

Russ, J. C. (2011). The image processing handbook. CRC Press, Boca Raton.

Russel, S., & Norvig, P. (2003). Artificial intelligence: A modern approach (M. J. Horton, Ed.). Prentice Hall Series in Artificial Intelligence.

Sandeep, A., & Parveen, L. (2012). Development of a seed analyzer using the techniques of computer vision. International Journal of Distributed and Parallel Systems (IJDPS), 3(1), 149-155.

Sattiraju, M., Manikantan, K., & Ramachandran, S. (2013, December). Adaptive bpso based feature selection and skin detection based background removal for enhanced face recognition. In Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2013 Fourth National Conference on IEEE., 1-4.

Scotland, R., & Wortley, A. (2003). How many species of seed plants are there? Taxon, 52, 101-104.

Seetha, M., Muralikrishna, I. V., Deekshatulu, I., Malleswari, B. L., & Nagaratna, P. H. (2008). Artificial neural networks and others methods of image processing classification. Journal of Theoretical and Applied Information Technology, 1039-1059.

Sergios, T., & Koutroumbas, K. (2010). An introduction to pattern recognition: A matlab approach. Academic Press imprint of Elsevier, 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA.

Shanmugapriya, D., & Padmavathi, G. (2013). A wrapper based feature subset selection using aco-elm-anp and ga-elm-anp approaches for keystroke dynamics authentication. In In signal processing image processing & pattern recognition (icsipr), 2013 international conference on (pp. 157-162). ieee.

Siddique, N., & Adeli, H. (2013). Computational intelligence: Synergies of fuzzy logic, neural networks, and evolutionary computing. John Wiley and Sons Ltd, The Atrium, Southern Gate, ChiChester, West Sussex, PO19 8SQ, United Kingdom.

Simon, X. L. (1993). Image analysis by moments. PhD thesis at the department of Electrical and Computer Engineering, The University of Manitoba Winnipeg, Manitoba, Canada.

Sivanandam, S. N., & Deepa, S. N. (2008). Introduction to genetic algorithms. Springer-Verlag , Berlin, Heidelberg.

Sobel, I. E. (1970). Cameral models and machine perception (Unpublished doctoral dissertation). Stanford University.

Somol, P., Baesens, B., Pudil, P., & Vanthienen, J. (2005). Filter - versus wrapper-based feature selection for credit scoring. International Journal of Intelligent Systems, 20(10), 985-999.

Specht, D. (1967). Generation of polynomial discriminant functions for pattern classification. IEEE Transactions on Electronic Computers,, 15, 3089.

Specht, D. (1971). Series estimation of a probability density function. Technometrics., 13, 4094.

Specht, D. (1990). Probabilistic neural networks and the polynomial adaline as complementary techniques for classification. IEEE Transactions on Neural Networks, 1, 1111.

Specht, D. F. (1988). Probabilistic neural networks for classification, mapping, or associative memory. IEEE International Conference on Neural Networks, 1(2), 525-532.

Stokes, M., Anderson, M., Chandrasekar, S., & Motta, R. (1996). A standard default color space for the internet - srgb. Microsoft and Hewlett-Packard, 1(10), 1-19.

Teague, M. (1980). Image analysis via the general theory of moments. J. Optical Soc. Am. 70, 920-930.

Teh, C. H., & Chin, R. T. (1988). On image analysis by the methods of moments. Pattern Analysis and Machine Intelligence, IEEE Transactions on,, 10(4), 496-513.

Thawar, A., Zyad Shaaban, K., Lala, & Sami, B. (2009). Object classification via geometric, zernike and legendre moments. Journal of Theoretical and Applied Information Technology, Vol 7. No 1, 31-37.

Tian, J., Hu, Q., Ma, X., & Ha, M. (2012). An improved kpca/ga-svm classication model for plant leaf disease recognition. Journal of Computational Information Systems, 18(8), 7737-7745.

Tuceryan, M., & Jain, A. K. (1998). Texture analysis :the handbook of pattern recognition and computer vision (2nd edition). World Scientific Publishing, 207-248.

Tyler, K. (2006). Fourier descriptors: Properties and utility in leaf classification. ECE 533 Fall 2006.

Umbaugh, S. E. (2011). Digital image processing and analysis. human and computer vision applications with cviptools. CRC Press: Taylor & Francis Group,Flourida, USA.

Urilch, B. S. L. K. B. . A. Z., W.; Peter. (2010). Plant species classication using a 3d lidar sensor and machine learning. Ninth International Conference on Machine Learning and Applications, 339-345.

Valliammal, N., & Geethalakshmi, S. N. (2011a). Automatic recognition system using preferential image segmentation for leaf and flower images. Computer Science & Engineering: An International Journal (CSEIJ), 1(4), 13-25.

Valliammal, N., & Geethalakshmi, S. N. (2011b). Hybrid image segmentation algorithm for leaf recognition and characterization. International Conference on Process Automation, Control and Computing (PACC), 1-6.

Van der Maaten, L. J. P., Postma, E. O., & Van Den Herik, H. J. (2009). Dimensionality reduction: A comparative review. Journal of Machine Learning Research, 10, 1-41.

Vorobyov, M. (2011). Shape classification using zernike moments. (Tech. Rep.). Technical Report. iCamp-University of California Irvine, 2011.

Wang, J. S., Song, J. D., & Gao, J. (2015). Rough set-probabilistic neural networks fault diagnosis method of polymerization kettle equipment based on shuffled frog leaping algorithm. Information., 6(1), 49-68.

Wang, K., Chen, T., & Lau, R. (2011). Bagging for robust non-linear multivariate calibration of spectroscopy. Chemometrics and Intelligent Laboratory Systems., 105(1), 1-6.

Wang, S., Jiliang, T., & Huan, L. (2015). Embedded unsupervised feature selection. Proceedings of the twenty-ninth AAAI Conference on Artificial Intelligence.

Wang, Y., Fan, X., & Cai, Y. (2014). A comparative study of improvements filter methods bring on feature selection using microarray data. Springer International Publisher, Switzerland, 55-62.

Wang, Z., Chi, Z., & Feng, D. a. (2003). Shape based leaf image retrieval. IEEE Proceedings(online 20030160).

Weight, C., Parnham, D., & Waites, R. (2008). Leafanalyser: a computational method for rapid and large scale analyses of leaf shape variation. TECHNICAL ADVANCE: The Plant Journal, 53(3), 578-586.

Whelan, M. A., B.M. (2003, Feb. 2-6). Definition and interpretation of potential management zones in australia. In: Proceedings of the 11th Australian Agronomy Conference. Geelong, Victoria.

Wu, S. G., Bao, F. S., Xu, E. Y., Wang, Y.-X., Chang, Y.-F., & Xiang, Q.-L. (2007). A leaf recognition algorithm for plant classification using probabilistic neural network. In Signal Processing and Information Technology, 2007 IEEE International Symposium., 11-16.

Xiao, Ji-Xiang, G., & Xiao-Feng, W. (2005). Leaf recognition based on the combination of wavelet transform and gaussian interpolation. China.

Xiao, Z., Dellandrea, E., Dou, W., & Chen, L. (2008). Esfs: A new embedded feature selection method based on sfs.