TOOL FOR IDENTIFYING CANCEROUS AND NON-CANCEROUS CELLS USING IMAGE PROCESSING AND ARTIFICIAL NEURAL NETWORKS

By

ANITYA NIJHARA-061554

SHANTANU AGARWAL-061288

DEPARTMENT OF COMPUTERS

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

MAY-2010

CERTIFICATE

This is to certify that the work entitled, "**A TOOL FOR DETECTION OF CANCEROUS AND NON-CANCEROUS CELLS USING IMAGE PROCESSING AND ARTIFICIAL NEURAL NETWORKS**" submitted by **Anitya Nijhara** and **Shantanu Agarwal** in partial fulfillment for the award of degree of Bachelor of Technology in Bioinformatics of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Nitin 19th May 2020

Dr. Nitin
(Project Supervisor)
Department of Computer Science
Jaypee University of Information Technology
Waknaghat

Naik 19/5/2010

Dr. Pradeep Kumar Naik
(Project supervisor)
Department of Bioinformatics
Jaypee University of Infomation Technology
Waknaghat

ACKNOWLEDGMENT

Many people have contributed to this project in a variety of ways over the past few months. To the individuals who have helped us, we again express our appreciation. We also acknowledge the many helpful comments Received from our teachers of the bioinformatics department. We are indebted to all those who provided reviews & suggestions for improving the results and the topics covered in our project, and we extend our apologies to anyone we may have failed to mention.

CONTENTS

CHAPTER 2: A TOOL FOR IDENTIFICATION OF CANCEROUS AND NON-CANCEROUS CELS USING IMAGE PROCESSING AND AI TECHNIQUES (ARTIFICIAL NEURAL NETWORK) ....28 - 50

LIST OF ABBREVIATIONS

ANN:	Artificial Neural Network.
DES:	Diethylstilbestrol
VAIN:	Vaginal Intraepithelial Neoplasia
HPV:	Human Papillomavirus
R:	Red
G:	Green
B:	Blue
N:	Nuclear size
C:	Cytoplasmic size
N/C:	Ratio of Nuclear size to Cytoplasmic size
ROC:	Receiver Output Characteristic

ABSTRACT

Motivation: Detection of cancerous squamousal cells in human beings prominently observed in females is a cumbersome process in the field of cancer diagnosis.Most of the currently used methods are manual and doesnot guaruantee accurate results.Here we present a method to predict both cancerous and non-cancerous squamousal cells using image processing and artificial neural networks.

Cancer is one of the principle causes of death in developed countries. The diagnosis of cancer is a tedious and cumbersome process, especially in the early stages. Hence our application, developed using image processing and artificial neural networks (ANN) for pattern recognition, automates this process to assist pathologists and lab technicians to achieve a more efficient and faster diagnosis.

Results: We have developed a tool based on two-layer neural network which classifies image of any given sqaumousal cell as cancerous or non-cancerous.
Using 15 derived features, the first layer of our neural network has been able to achieve 66% correct detection of nucleus, cytoplasm and background. Finally, the tool is able to classify the images into cancerous/ non-cancerous with an accuracy of 69%. For the complete set of 15 parameters using 5 fold cross validation classification, our ANN model reveals an accuracy of 69.18±6.86%, a sensitivity of 78.96±6.73%, a specificity of 97.86±13.39% and MCC 64.93±6.98. This shows that computer aided diagnosis can be a helpful tool, especially in a field that lacks experienced specialists.

CHAPTER 1

INTRODUCTION

Cancer of the vagina, a rare kind of cancer in women, is a disease in which malignant cells are found in the tissues of the vagina. There are several types of cancer of the vagina. The most common is

- squamous cell cancer (squamous carcinoma)
    - Squamous carcinoma is most often found in women between the ages of 60 and 80, and accounts for 85-90 percent of all vaginal cancers.

COMMANALITY OF CANCER

It is a rare cancer - only about 200 new cases were diagnosed in the UK in 2006. Cancer starting in another place in the body such as cancer of the cervix, womb cancer or bowel cancer can spread to the vagina. This is not the same as cancer starting in the vagina. Cancer starting in the vagina is known as primary vaginal cancer. Cancer that has spread from another place in the body is called secondary cancer. Because vaginal cancer is such a rare type of cancer, it is very difficult to carry out research involving large enough numbers to give any reliable results. But researchers have managed to identify several risk factors.

RISK FACTORS

The following have been suggested as risk factors for vaginal cancer:

- Age, half of women affected are older than 60, with most between ages 50 and 70.
- Exposure to diethylstilbestrol (DES) as a fetus (mother took DES during pregnancy)
- History of cervical cancer
- History of cervical precancerous conditions
- Human papillomavirus (HPV) infection
- Vaginal adenosis
- Vaginal irritation
- Uterine prolapse
- Smoking

SYMPTOMS

It is rare to have symptoms if you have very early stage vaginal cancer or the pre-cancerous changes called vaginal intraepithelial neoplasia (VAIN). As with most cancers, this early stage disease is easy to treat successfully.

Although some early stage vaginal cancers may have symptoms, many do not until they are in the advanced stages. As many as 2 women in 10 (20%) diagnosed with vaginal cancer don't have symptoms at all. This is most likely to be because their cancer has been found 'by accident' at an early stage – probably during a routine examination or smear.

Possible symptoms

Overall, between 8 and 9 out of 10 women (80 – 90%) have one or more symptoms with vaginal cancer, including

- Bleeding when you are not having a period or after menopause. As many as 8 out of 10 women have this symptom with vaginal cancer. You may have bleeding after sex
- Vaginal discharge that smells or may be blood stained – about 3 out of 10 women have this symptom
- Pain during sexual intercourse
- A lump or growth in the vagina that you or your doctor can feel – up to 1 in 10 women has this
- A vaginal itch that won't go away

Many of these symptoms can be caused by other conditions, such as infections.

Other symptoms

As well as the above, the following symptoms are more likely with advanced cancer of the vagina

- Constipation
- Pain when passing urine
- Swelling in your legs (oedema)
- Pain in the pelvic area that won't go away

IMAGE PROCESSING

Image processing is any form of information processing for which the input is an image, such as photographs or frames of video; the output is not necessarily an image, but can be for instance a set of features of the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it.

Techniques include convolution edge detection, mathematics, filters, trend removal, and image analysis. The various image enhancements and image processing techniques will be introduced in this section. Computer software programs are available, including some or all of the following programs:

Enhancement programs make information more visible.

- Histogram equalization-Redistributes the intensities of the image of the entire range of possible intensities (usually 256 gray-scale levels).
- Unsharp masking-Subtracts smoothed image from the original image to emphasize intensity changes.

Convolution programs are 3-by-3 masks operating on pixel neighborhoods.

- Highpass filter-Emphasizes regions with rapid intensity changes.
- Lowpass filter-Smoothes images, blurs regions with rapid changes.

Math processes programs perform a variety of functions.

- Add images-Adds two images together, pixel-by-pixel.
- Subtract images-Subtracts second image from first image, pixel by pixel.
- Exponential or logarithm-Raises e to power of pixel intensity or takes log of pixel intensity. Nonlinearly accentuates or diminishes intensity variation over the image.

- Scaler add, subtract, multiply, or divide-Applies the same constant values as specified by the user to all pixels, one at a time. Scales pixel intensities uniformly or non-uniformly
- Dilation-Morphological operation expanding bright regions of image.
- Erosion-Morphological operation shrinking bright regions of image.

Noise filters decrease noise by diminishing statistical deviations.

- Adaptive smoothing filter-Sets pixel intensity to a value somewhere between original value and mean value corrected by degree of noisiness. Good for decreasing statistical, especially single-dependent noise.
- Median filter-Sets pixel intensity equal to median intensity of pixels in neighborhood. An excellent filter for eliminating intensity spikes.
- Sigma filter-Sets pixel intensity equal to mean of intensities in neighborhood within two of the mean. Good filter for signal-independent noise.

Trend removal programs remove intensity trends varying slowly over the image.

- Row-column fit-Fits image intensity along a row or column by a polynomial and subtract fit from data. Chooses row or column according to direction that has the least abrupt changes.

Edge detection programs sharpen intensity-transition regions.

- First difference-Subtracts intensities of adjacent pixels. Emphasizes noise as well as desired changes.
- Sobel operator-3-by-3 mask weighs inner pixels twice as heavily as corner values. Calculates intensity differences.
- Morphological edge detection-Finds the difference between dilated (expanded) and eroded (shrunken) version of image.

Image analysis programs extract information from an image.

- Gray-scale mapping-Alters mapping of intensity of pixels in file to intensity displayed on a computer screen.
- Slice-Plots intensity versus position for horizontal, vertical, or arbitrary direction. Lists intensity versus pixel location from any point along the slice.
- Image extraction-Extracts a portion or all of an image and creates a new image with the selected area.
- Images statistics-Calculates the maximum, minimum, average, standard deviation, variance, median, and mean-square intensities of the image data.

MACHINE LEARNING TECHNIQUES

NEURAL NETWORKS:

Neural Network or more appropriately Artificial Neural Network is basically a mathematical model of what goes in our mind (or brain). The brain of all the advanced living creatures consists of neurons, a basic cell, which when interconnected produces what we call Neural Network. The sole purpose of a Neuron is to receive electrical signals, accumulate them and see further if they are strong enough to pass forward.
The basic functionality lies not in neurons but the complex pattern in which they are interconnected. NNs are just like a game of chess, easy to learn but hard to master. In the same way, a single neuron is useless. Well, practically useless. It is the complex connection between them and values attached with them which makes brains capable of thinking and having a sense of consciousness (much debated).

Basic working principle of a neuron

A neuron is basically a cell which accumulates electrical signals with different strengths. What it does more is that it compares the accumulated signal with one predefined value unique to every neuron. This value is called bias. Function of a neuron could be explained in the following diagram.

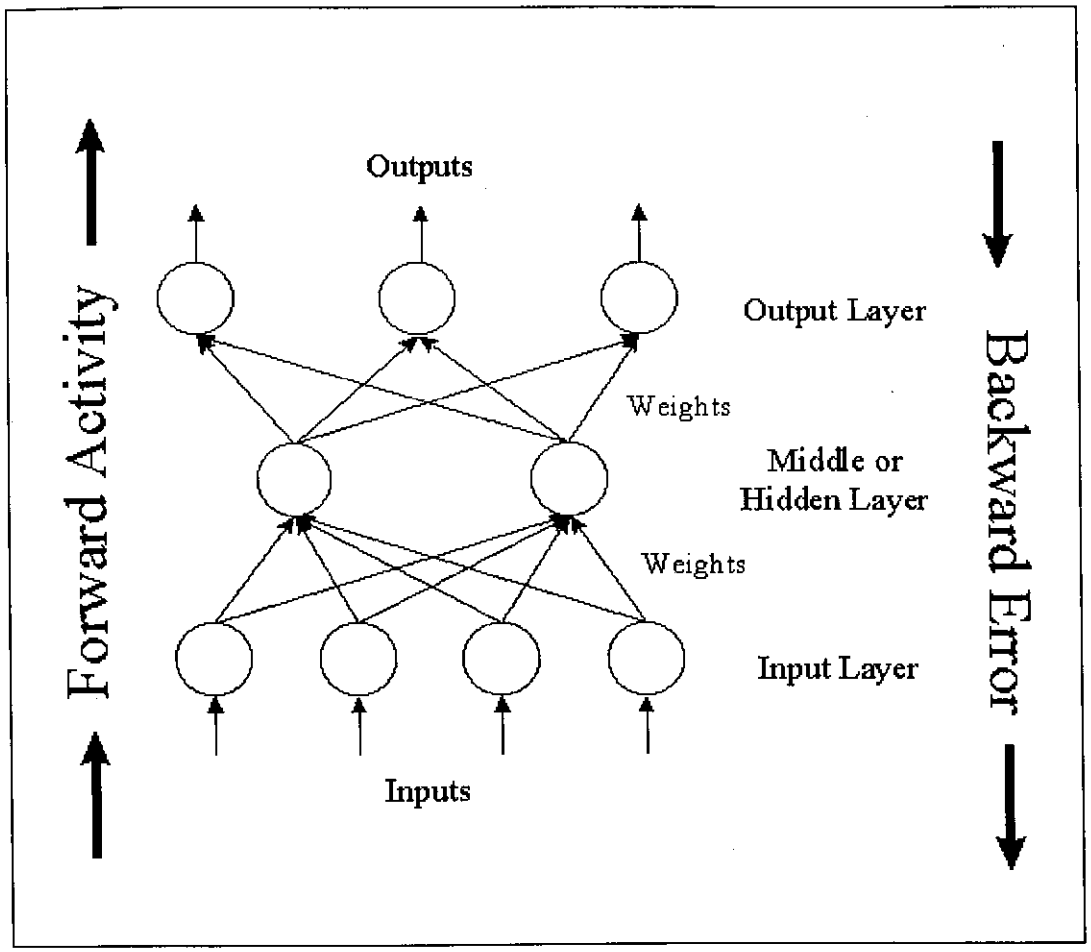


FIGURE 4.1. Typical Artificial Neural Network Setup (Caudill and Butler, 1992a).

Image Source: http://www.interwet.psu.edu/f41.gif

The circles in the image represent neurons. This network or more appropriately this network topology is called feed-forward multi layered neural network. It is the most basic and most widely used network. The network is called multi layered because it consists of more than two

layers. The neurons are arranged in a number of layers, generally three. They are input, hidden/middle and output layers.

This network is feed-forward, means the values are propagated in one direction only. There are many other topologies in which values can be looped or move in both forward and backward direction. But, this network allows the movement of values only from input layer to output layer. The functions of various layers are explained below:

Input layer: As it says, this layer takes the inputs and forwards it to hidden layer. We can imagine input layer as a group of neurons whose sole task is to pass the numeric inputs to the next level. The larger the number greater its strength. E.g. 0.51 is stronger than 0.39 but 0.93412 is stronger still. But, the interpretation of this strength depends upon the implementation and the type of problem assigned to NN to solve. For an OCR you connect every pixel with its respective input neuron and darker the pixel, higher the signal/input strength. Input layer never processes data, it just hands over it.

Middle layer: This layer is the real thing behind the network. Without this layer, network would not be capable of solving complex problems. There can be any number or middle or hidden layers. But, for most of the tasks, one is sufficient. The number of neurons in this layer is crucial. This layer takes the input from input layer, does some calculations and forwards to the next layer, in most cases it is the output layer. There is no specifc formula for deciding the number of hidden nodes.

Output layer: This layer consists of neurons which preditct the output value of the given input data. This layer takes the value from the previous layer, **does calculations** and gives the final result. Basically, this layer is just like hidden layer but instead of passing values to the next layer, the values are treated as output.

Dendrites: These are straight lines joining two neurons of consecutive layers. They are just a passage (or method) through which values are passed from one layer to the next. There is a value attached with dendrite called **weight**. The weight associated with dendrites basically determines the importance of incoming value. A weight with larger value determines that the value from that particular neuron is of higher significance. To achieve this we do is multiply the incoming value

with weight. So no matter how high the value is, if the weight is low the multiplication yields the final low value.

Training: Training is the most important part of a neural network and the one consisting of the most mathematics. It uses Backpropagation method for training the NN. The best example illustrating this principle is Charles Darwin(what?). Yes, at the time when he wrote '*On the Origin of Species*', DNA was not known. So, he propounded the evolution without even knowing the method of how it is done i.e. how traits are passed on from parents to offspring. Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimises the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation. Most of the algorithms used in training artificial neural networks are employing some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction. Evolutionary methods, simulated annealing, and expectation-maximization and non-parametric methods are among other commonly used methods for training neural networks. This training procedure must be repeated for larger number of samples so that the NN can produce accurate results for untrained input samples.

Applications: The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical.

Real life applications

The tasks to which artificial neural networks are applied tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction and modeling.

- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.

Application areas include system identification and control (vehicle control, process control), game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition and more), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications, data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering.

References

- National Cancer Institute: Vaginal Cancer (public domain)
- [Stenchever: Comprehensive Gynecology, 4th ed., Copyright © 2001 Mosby, Inc.]
- *The Image Processing Handbook* by John C. Russ, ISBN 0849372542 (2006)
- *Fundamentals of Image Processing* by Ian T. Young, Jan J. Gerbrands, Lucas J. Van Vliet, Paperback, ISBN 90-75691-01-7 (1995)
- *Image Analysis and Mathematical Morphology by Jean Serra, ISBN 0126372403 (1982)*
- *Front-End Vision and Multi-Scale Image Analysis* by Bart M. ter Haar Romeny, Paperback, ISBN 1-4020-1507-0 (2003)
- Christopher M. Bishop (2007) *Pattern Recognition and Machine Learning*, Springer ISBN 0-387-31073-8.
- *Neural Computing and Applications*, Springer-Verlag. (address: Sweetapple Ho, Catteshall Rd., Godalming, GU7 3DJ)

- Bhagat, P.M. (2005) *Pattern Recognition in Industry*, Elsevier. ISBN 0-08-044538-1

- Bishop, C.M. (1995) *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press. ISBN 0-19-853849-9 (hardback) or ISBN 0-19-853864-2 (paperback)

- Duda, R.O., Hart, P.E., Stork, D.G. (2001) *Pattern classification (2nd edition)*, Wiley, ISBN 0-471-05669-3

- Gurney, K. (1997) *An Introduction to Neural Networks* London: Routledge. ISBN 1-85728-673-1 (hardback) or ISBN 1-85728-503-4 (paperback)

- Haykin, S. (1999) *Neural Networks: A Comprehensive Foundation*, Prentice Hall, ISBN 0-13-273350-1

Objectives

With the increase in the number of cases of squamousal cancer in human beings predominantly in females, accurate prediction of the disesase is very much needed.Since,the manual methods are time consuming and can not encompass all the boundaries to visualize an image,there is an increasing demand for the development of a machine or tool that can predict the cancerous cells more efficiently and accurately. Hence, in this study an attempt has been taken to develop an automated tool using image processing and artificial neural networks for detection of cancerous and non-cancerous squamousal cells with the following objectives:

1. To obtain the distinct images of human sqaumousal cancerous and non-cancerous cells.

2. To extract statistical parameters of certain pixels from nuclear, cytoplasmic, background portions of the images.

3. To develop a neural network(1^{st} layer) for identification of nuclear,cytoplasmic,background portions of any given image.

4. To develop a neural network(2^{nd} layer)for the prediction and classification of any given squamousal image based on N,C,and N/C parameters.

CHAPTER 2

A tool for Identification and classification of squamousal cancerous and non-cancerous cells using Image Processing and Artificial Neural Networks.

1 INTRODUCTION:

Cancer is a class of diseases or disorders characterized by uncontrolled division of cells and the ability of these cells to invade other tissues, either by direct growth into adjacent tissue through *invasion* or by implantation into distant sites by *metastasis*. Metastasis is defined as the stage in which cancer cells are transported through the bloodstream or lymphatic system. Cancer may affect people at all ages, but risk tends to increase with age, due to the fact that DNA damage becomes more apparent in aging DNA. It is one of the principal causes of death in developed countries.Here ,we are dealing with the cancer of the vagina, a rare kind of cancer in women, is a disease in which malignant cells are found in the tissues of the vagina. Because vaginal cancer is such a rare type of cancer, it is very difficult to carry out manual research involving large enough numbers to give any reliable results.

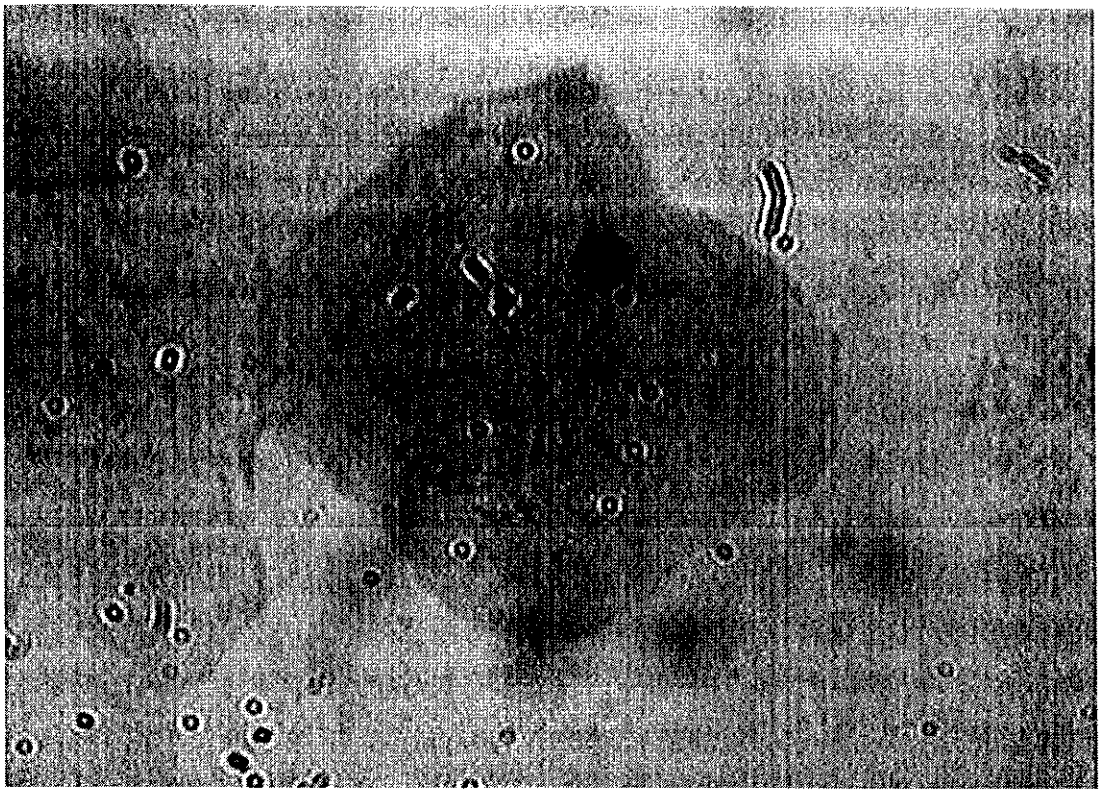
Image Processing and Artificial Neural Networks have been applied for detection and identification of squamousal cancerous and non-cancerous cells in human.

2 MATERIALS AND METHODS:

2.1 Images (Training Data)

The training data set constists of pathologiocal images of cancerous and non-cancerous human squamousal cells. This data set was obtained from AIIMS, New Delhi.

IMAGE OF NON-CANCEROUS SQUAMOUSAL CELLS (Training Data)

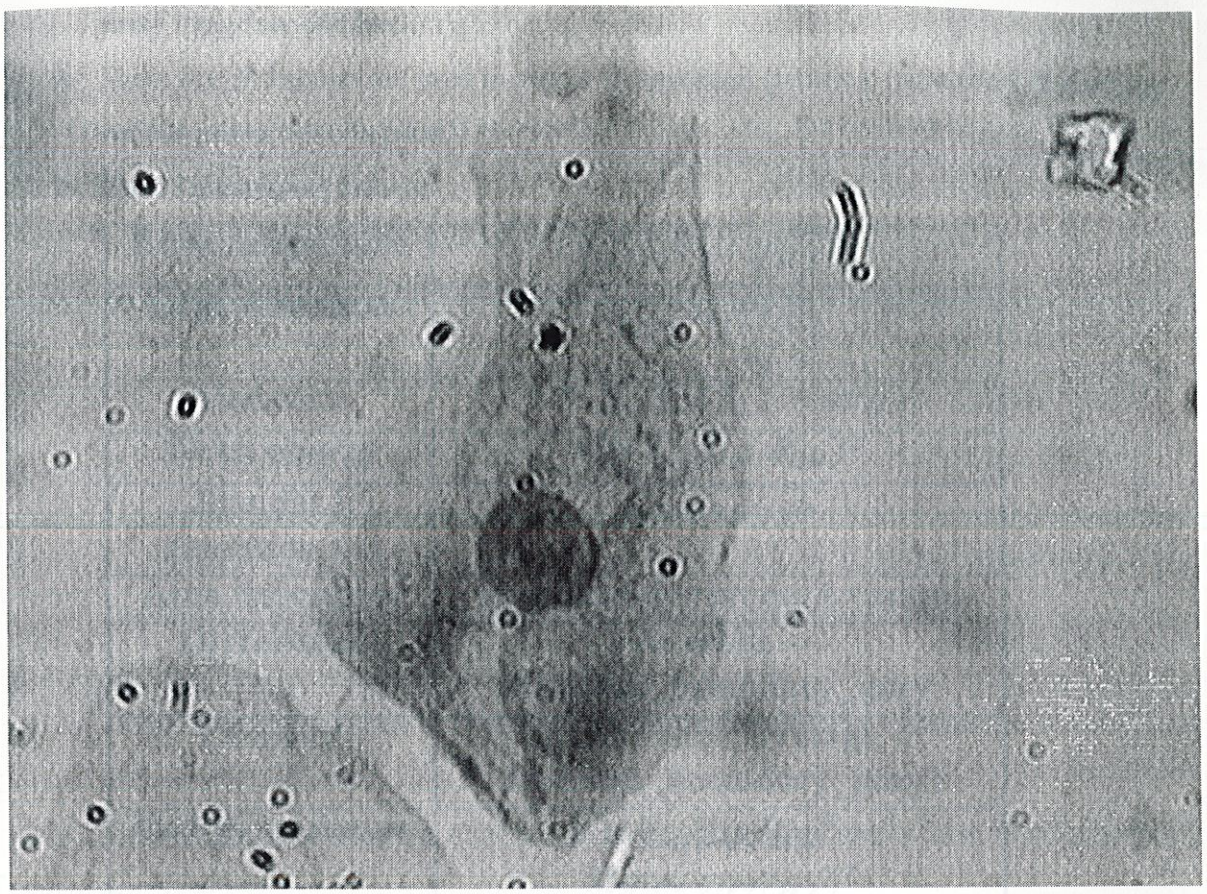


Typical squamousal cells of undetermined significance as seen on a large parabasal cell (Papanicolaou, medium power)

General features of normal **non-cancerous** squamousal cells are:

- The nucleus of a non-cancerous cell is small in size.
- The surface of a non-cancerous cell is smooth and regular.
- The ratio of nuclear volume to its cellular volume in a non-cancerous cell is low.

IMAGE OF CANCEROUS SQUAMOUSAL CELLS (Training Data)



Cell spread from a large cell invasive squamousal carcinoma. The cells tend to form irregular clusters. Anisokaryosis is marked. The nuclei are hyperchromatic with coarse chromatin. The nuclear membrane seems irregularly thickened (Papanicolaou, medium power).

General features of normal **cancerous** squamousal cells are:

- The size of nucleus of a cancerous cell is large compared to that of a Non cancerous cell.
- The surface of a cancerous cell is generally very rough and irregular whereas that of a non-cancerous cell is comparatively smooth and regular.
- The ratio of nuclear volume to its cellular volume in a cancerous cell is high.

2.2 Features Extraction

- R,G,B features of selected co-ordinates from nuclear,cytoplasmic,background parts of every image are obtained.
- These R,G,B features are used to obtain 15 important statistical parameters describing various properties of a pixel.
- The 15 parameters include:

Edge magnitude	$(\sqrt{dx^2} + \sqrt{dy^2})$
Edge direction	$tan^{-1}(\frac{dfy}{dfx})$
Variance	$1/N \sum_u \sum_v (f_{u,v} - avg)$
Mean energy	$1/N \sum_u \sum_v (P_{u,v})^2$
Standard deviation (σ)	$\sqrt{variance}$
Area descriptor	Standard deviation/Average
Difference	Average – Average_boundary_gray
Contrast	Difference/(Average + Average_boundary_gray)
Energy variance	$1/N \sum_u \sum_v (P_{u,v}{}^2 - Mean\ energy)^2$
Maximum	The highest value of pixels out of 3x3 matrix
Minimum	The lowest value of pixels out of 3x3 matrix
Average	The average value of pixels out of 3x3 matrix
Red	Intensity of red color of a pixel
Green	Intensity of green color of a pixel
Blue	Intensity of blue color of a pixel

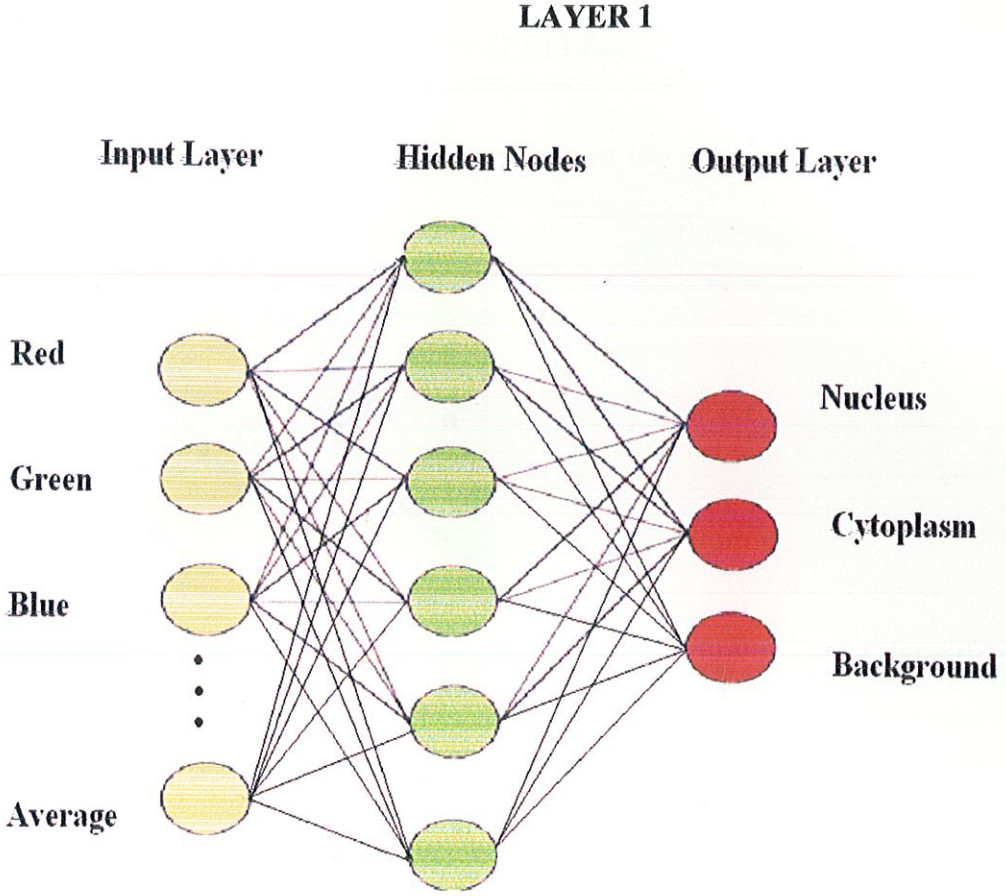
- The 15 statisitical parameters thus obtained for pixels of nuclear, cytoplasmic, background parts are used as an input to train the 1st level of neural network for identifying the corresponding category of any given pixel.

2.3 Parameters (N,C,N/C)

- The 1st layer of neural network is trained and is used to identify various parts of an image.
- Various images of squamousal cancerous and non-cancerous cells are parsed through the trained network for the identification of nuclear, cytoplasmic and background parts.
- The output obtained from the above process is used for the calculation of three parameters namely N, C, N/C serving the input for neural network layer2.
- The above parameters namely N, C, N/C determine the nuclear volume, cytoplasmic volume and the ratio of size of the nucleus to its cytoplasm in a given image
- Thereby, the above three parameters for squamousal cancerous and non-cancerous cells are obtained by using neural network layer1.
- These serve the input for 2nd layer of neural network where it classifies a given image as cancerous or non-cancerous.
- This 2nd layer of neural network is now hereby trained for the above mentioned classification.

N	**Nuclear volume**
C	**Cytoplasmic volume**
N/C	**Ratio of size of nucleus to cytoplasm in a given image**

2.4 ANN Model



The first layer of ANN consists of 15 input nodes, one for each feature mentioned in Table 2, and 3 output nodes, one for each region mentioned in Figure 3. The number of hidden nodes was varied and the network was found to be optimized with 7 hidden nodes. The normalized pixel feature values were given as input and the network was trained to recognize and classify the pixels into nuclear, cytoplasmic or background pixels.

TRAINING OF THE FIRST LAYER

The 15 feature values of a pixel are fed as input and a random output is generated through the untrained network. We compare this output with the desired output value and backpropogate the error, if any. We repeat this process for each of the 100 selected pixels of each region in each image of the training data set. Now our first layer can identify different regions of an image.

INTERMEDIATE STEP

Once the first layer is successfully trained, we run it on 100 randomly selected pixels from a particular image.

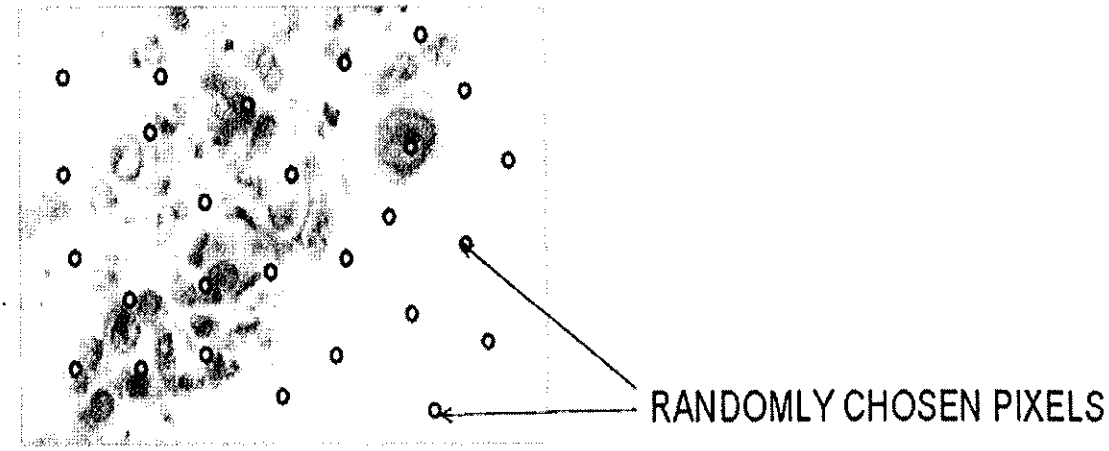
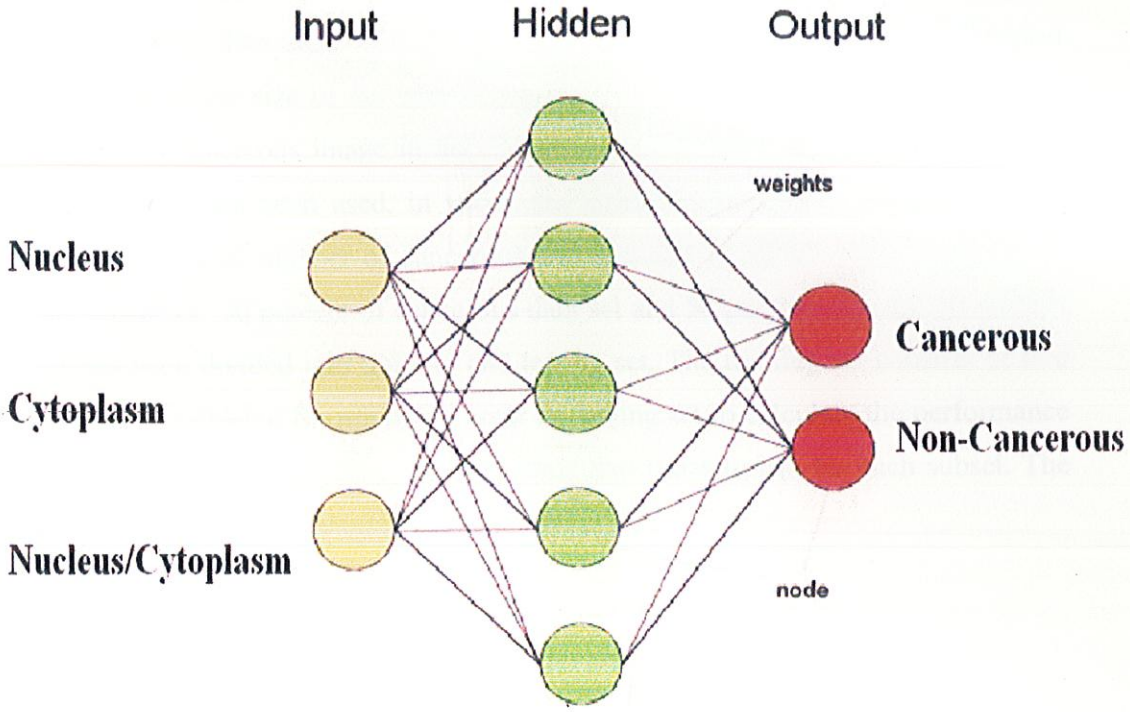


Figure 2.Randomly selected pixels to accumulate image wise data in the intermediate step.

Since the selection of these pixels is completely random, probabilistic logic demands that the ratio of random pixels picked up from all the three regions will roughly correspond to the ratio of areas covered by them in the image. This information, gathered from the first layer is used to calculate the three parameters (Table 3) that are used to train the second layer. So, as an intermediate step between the two layers, we accumulate the pixel level results of one image and calculate the image level parameters.



The second layer consists of 3 input nodes, one for each parameter mentioned in Table 3, and 2 output nodes, for the final classification. The number of hidden nodes was varied and the network was found to be optimized at 19.

TRAINING OF THE SECOND LAYER

Since the primary difference between cancerous and non cancerous squamousal cells (as mentioned earlier) lies in the relative sizes of the nucleus and cytoplasm, we use the ratio of the volume of the Nucleus the Cytoplasm as a basis for training the second layer to identify cancerous images from non cancerous ones. The parameter values were given as input and the network was trained to recognize and classify the images as cancerous or non-cancerous.

2.5 Five-fold cross validation

A prediction method is often developed by cross-validation or jack-knife method (Chou and Zhang, 1995). Because of the size of the dataset, the jack-knife method (individual testing of each cancerous and non-cancerous image in the data set) was not feasible. So a more limited cross-validation technique has been used, in which the dataset is randomly divided into five subsets, each containing equal number of cancerous and non-cancerous data sets.Each set is a balanced set that consist of 50 percent of cancerous data set and 50 percent non-cancerous data set. The data set has been divided into training and testing set. The training set consists of five subsets. The network is validated for minimum error on testing set to calculate the performance measure for each fold of validation. This has been done five times to test for each subset. The final prediction results have been averaged over five testing sets.

2.6 Performance Measure

The prediction results of ANN model developed in the study were evaluated using the following statistical measures.

1. *Accuracy of the methods*: The accuracy of prediction for neural network models were calculated as follows:

$$Q_{ACC} = \frac{P+N}{T}, \text{ where T = (P+N+O+U)}$$

Where P and N refer to correctly predicted enzymes and non-enzymes, and O and U refer to over and under predictions, respectively.

2. The Matthews correlation coefficient (MCC) is defined as:

$$MCC = \frac{(P\times N)-(O\times U)}{\sqrt{(P+U)\times(P+O)\times(N+U)\times(N+O)}}$$

3. Sensitivity (Q_{sens}) and specificity (Q_{spec}) of the prediction methods are defined as:

$$Q_{sens} = \frac{P}{P+U}$$

$$Q_{spec} = \frac{N}{N+O}$$

4. Q_{Pred} (Probability of correct prediction) and Q_{obs} (Percentage over coverage) are defined as:

$$Q_{pred} = \frac{P}{P+O} \times 100$$

$$Q_{obs} = \frac{P}{P+U} \times 100$$

RESULTS

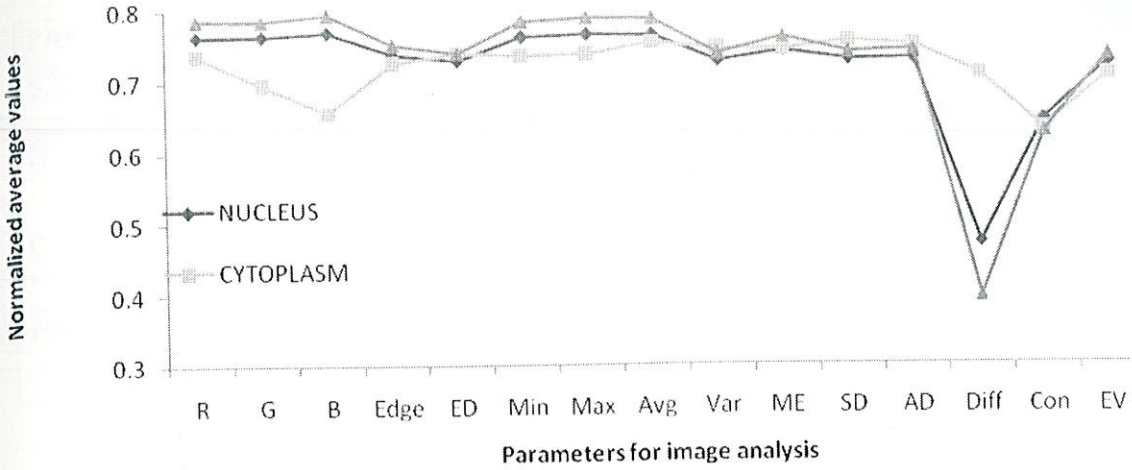
15 parameters calculated from the CANCEROUS image with respect to nucleus, cytoplasm and background:

PARAMETERS	NUCLEUS			CYTOPLASM			BACKGROUND		
	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
R	0.400000	0.790053	0.763226	0.400000	0.847188	0.736430	0.400000	0.847188	0.786171
G	0.400000	0.787969	0.761971	0.400000	0.844369	0.693677	0.228741	0.844369	0.784492
B	0.400000	0.787135	0.767783	0.400000	0.843242	0.655340	0.061165	0.843242	0.792272
Edge magnitude	0.400000	0.851744	0.736245	0.400000	0.930607	0.723193	0.400000	0.931623	0.750056
Edge direction	0.400000	0.892849	0.727957	0.400000	0.986189	0.738693	0.136861	1.000000	0.738963
Minimum	0.400000	0.783521	0.760418	0.400000	0.838356	0.735018	0.024445	0.838356	0.782413
Maximum	0.400000	0.791721	0.764296	0.400000	0.849443	0.735899	0.400000	0.849443	0.787602
Average	0.400000	0.787845	0.762326	0.400000	0.844202	0.752947	0.400000	0.844202	0.784967
Variance	0.400000	0.727343	0.725872	0.400000	0.762392	0.743021	0.374744	0.762392	0.736161
Mean Energy	0.400000	0.762073	0.740003	0.400000	0.809354	0.741498	0.400000	0.777430	0.759454
Standard deviation	0.400000	0.738464	0.727090	0.400000	0.777430	0.754146	0.400000	0.809354	0.737801
Area descriptor	0.400000	0.789013	0.730011	0.400000	0.845782	0.749390	0.400000	0.845782	0.741712
Difference	0.291697	0.725586	0.470406	0.173309	0.760016	0.708304	0.172764	0.760016	0.394214
Contrast	0.400000	0.643455	0.643145	0.400000	0.648957	0.631171	0.000000	0.648957	0.625436
Energy	0.400000	0.727001	0.725864	0.400000	0.761928	0.708878	0.400000	0.761928	0.736147

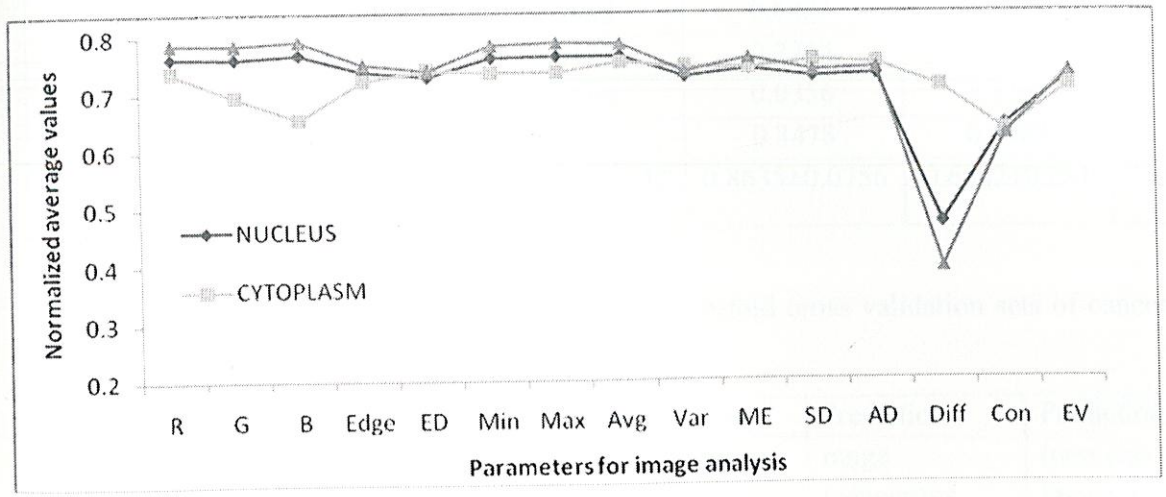
15 parameters calculated from the NON-CANCEROUS image with respect to nucleus, cytoplasm and background:

PARAMETERS	NUCLEUS			CYTOPLASM			BACKGROUND		
	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
R	0.400000	0.808743	0.768703	0.400000	1.000000	0.738074	0.400000	1.000000	0.786171
G	0.400000	0.807808	0.769891	0.023906	0.901296	0.711774	0.011975	0.901296	0.700897
B	0.400000	0.803600	0.767232	0.400000	0.803600	0.740131	0.400000	0.803600	0.610623
Edge magnitude	0.400000	1.000000	0.718060	0.400000	1.000000	0.707324	0.400000	1.000000	0.750056
Edge direction	0.400000	0.889417	0.703478	0.400000	0.901296	0.696850	0.349590	0.901267	0.664845
Minimum	0.400000	0.788951	0.762039	0.150973	0.788951	0.732129	0.150973	0.788951	0.675152
Maximum	0.400000	0.817937	0.768359	0.400000	0.817937	0.743531	0.400000	0.817937	0.675544
Average	0.400000	0.802907	0.765318	0.400000	0.802907	0.732862	0.400000	0.802907	0.698570
Variance	0.400000	0.711434	0.702283	0.400000	0.901296	0.681888	0.348545	0.901296	0.658059
Mean Energy	0.400000	0.788081	0.737711	0.009124	0.788081	0.620628	0.009124	0.788081	0.624359
Standard deviation	0.400000	0.735422	0.704231	0.400000	0.762892	0.663297	0.400000	0.762892	0.655057
Area descriptor	0.400000	0.834311	0.706947	0.000000	0.834311	0.687740	0.000000	0.834311	0.671827
Difference	0.400000	0.701001	0.260109	0.173309	0.756653	0.338858	0.400000	0.756653	0.411167
Contrast	0.400000	0.611734	0.609429	0.400000	0.756653	0.617782	0.400000	0.648957	0.608188
Energy	0.400000	0.707989	0.702245	0.400000	0.756049	0.694505	0.400000	0.756049	0.665733

Graph obtained for 15 parameters calculated from the CANCEROUS image with respect to nucleus, cytoplasm and background:



Graph obtained for 15 parameters calculated from the NON-CANCEROUS image with respect to nucleus, cytoplasm and background:



The parameters are scaled down by appropriate scaling values.

PARAMETERS	CANCEROUS			NON-CANCEROUS		
S.NO.	Max.	Min.	Avg.	Max.	Min.	Avg.
N	0.003916	0.999961	0.941952	0.003916	0.637982	0.321458
C	0.000039	0.009184	0.001353	0.000000	0.009184	0.001589
N/C	0.000284	0.031186	0.009735	0.000284	0.049938	0.000873

Results of cancerous/non- prediction methods, using five-fold cross validation:

5-fold cross validation	Accuracy	Specificity	Sensitivity	MCC	Q(Pred)
Using image derived features					
C1	0.6879	0.8765	0.8761	0.8214	89.62
C2	0.6796	0.7988	0.8421	0.5876	76.53
C3	0.7123	0.9123	0.7754	0.4445	72.43
C4	0.7054	0.8954	0.9356	0.5235	63.76
C5	0.6859	0.9983	0.8478	0.5769	71.23
Mean	0.6923±0.0586	0.9786±0.1627	0.8635±0.0756	0.6542±0.281	75.657±15.189

Output values from the neural network for the fivefold cross validation sets of cancerous/non-cancerous images:

Testing 5-fold cross validation	Number of cancerous images correctly predicted (out of 8)	Number of non-cancerous images correctly predicted (out of 9)	Prediction range (cancerous images)	Prediction range (non-cancerous images)
Using image features				
C1	7	4	0.8859-0.9876	0.7896-0.9876
C2	7	3	0.8967-0.9754	0.6743-0.9758
C3	6	5	0.8763-0.9789	0.8076-0.9632
C4	4	3	0.9015-0.9246	0.6754-0.8586
C5	8	2	0.9136-0.9469	0.6686-0.8236

Three parameters used as input for second layer ANN.

Parameters	CANCEROUS			NON-CANCEROUS		
S.No.	Max.	Min.	Avg.	Max.	Min.	Avg.
N	0.003916	0.999961	0.941952	0.003916	0.637982	0.321458
C	0.000039	0.009184	0.001353	0.000000	0.009184	0.001589
N/C	0.000284	0.031186	0.009735	0.000284	0.049938	0.000873

EVALUATION OF PREDICTION ACCURACY

A trade-off was observed between making few false-positive predictions and having a high sensitivity, which is, correctly identifying as many positive examples as possible. This can be visualized as what is known as the receiver output characteristic (ROC) curve, in which the sensitivity is plotted as a function of the 1-specificity by varying the score threshold used for making positive predictions. Figure shows the ROC curves for the two predictors included in our method. Performance of both networks has been evaluated by calculating the area under the ROC curve. The area under the curve is 0.88 for second layer of ANN showing better discrimination of cancerous and non-cancerous images.

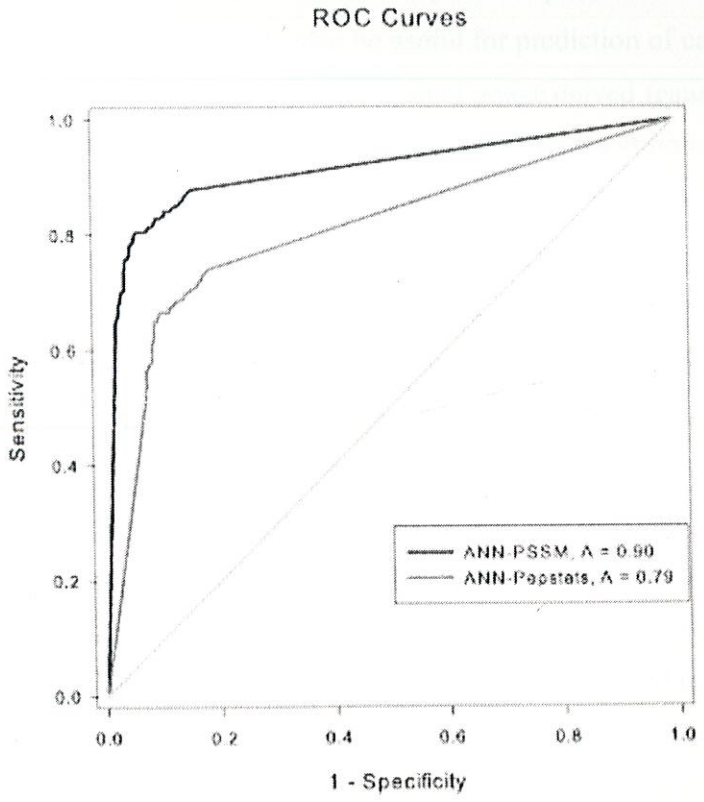


Figure: ROC curves for two different network systems

3 DISCUSSION

The tool has been developed in this study using two layered neural network based on image derived features. The results demonstrate that the developed ANN-based model for binary prediction of cancerous and non-cancerous images is adequate and can be considered an effective tool for cancer diagnosis. The results also demonstrate that the image derived parameters readily accessible from the digital images only, can produce a variety of useful information to be used in cancer diagnosis; clearly demonstrates an adequacy and good predictive power of the developed ANN model. There is strong evidence, that the introduced image features do adequately reflect the various parameters of an image. The statistical parameters of an image, the nuclear size and shape are important for the detailed description of an image, and wouldconsequently also be useful for prediction of cancerous/non-cancerous images. Based on the analysis of limited image deived featuires from squamousal cell images, differences in the parameters between cancerous and non cancerous images have previously been shown to exist and used for prediction of the same. This agrees well with our result that image derived features can be used for predicting cancerous and non cancerous images. This observation is not surprising considering that the calculated parameters should cover a very broad range of properties such as R, G, B, Minimum, Maximum, Average, Standard deviation, Energy variance etc. As it can be seen that the average value for all the nuclear, cytoplasmic and background portions were clearly separated on the graph and, hence, the selected 15 parameters should allow building an effective ANN model (1st layer) for proper identification of various parts of the image using which further the image is classified as cancerous or non cancerous using the 2nd layer ANN model. The average value of all the image derived parameters used in the study for all the squamousal images are clearly separated and could be suitable for classification.

The ANN with 15 input nodes and 3 output nodes has allowed the recognition of >95% of nuclear, cytoplasmic and background portions, on average and has also demonstrated very good separation. The second layer of neural network with 3 input nodes and 2 output nodes is able to correctly classify 80% of the cancerous images. This result revealed a good prediction with accuracy.

Presumably, the accuracy of the approach operating by the image derived features can be improved even further by expanding the parameters or by applying more powerful classification techniques such as Support Vector Machines or Bayesian Neural Networks. Use of merely statistical techniques in conjunction with the sequence parameters would also be beneficial, as they will allow interpreting individual parameter contributions into "cancerous/non-cancerous image likeness".

The results of the present work demonstrate that the image derived features with ANN appear to be a very fast image identification mechanism providing good results, comparable to some of the current efforts in the literature. The developed ANN-based model for cancerous/non-cancerous image prediction can be used as a powerful tool for filtering through the various squamousal cell images.

4 CONCLUSIONS

Froma practical point of view the most important aspect of a prediction method is its ability to make correct predictions. Till date there is no automated tool available for the identification of cancerous, non-cancerous squamousal cell images. This is a very tedious job and requires much costlier endeavors. The statistically derived features from images are important determinants for the detailed identification of squamousal cell images. Therefore, a much accurate and reliable method is that which predicts the cancerous and non-cancerous images classes based on the above mentioned strategy.

This thesis contains detailed workon image classification. The neural network architecture used for the prediction was optimized for maximum accuracy. This was done by gradually testing networks with variable hidden nodes and retaining the one with the highest truest predictions. This is at par with the best prediction tools available till date, but to the contrary, uses a simple and efficient prediction mechanism based on image derived features. This application not only gives optimum result with the dataset used, but also predicts the low quality images to a very high satisfactory level.

The second layer of ANN model (3-19-2) is trained with the image derived features (3 parameters) obtained from first layer of ANN (15-7-3) which uses 15 features derived from images as input. When applying a fivefold cross-validation test using five data sets, we found that the network reached an overall accuracy of 69.18±5.86%. The prediction results are presented in Table 6. The net has achieved an MCC of 0.649349±0.281. The other performance measures are sensitivity = 88.96±7.56% and specificity = 97.86±16.27%. Training was performed for 100 epochs for both the networks, after which the learning has been terminated when the error reached a sufficiently stable value.

Out of 8 in each cross-validation set 4-8 cancerous images were correctly predicted as cancerous. However, out of 9 in non-cancerous images; 2-5 were correctly predicted as non-cancerous. Prediction performance measures were averaged over five sets. This illustrates that maximum of the predicted values provide very adequate separation of the two classes of images using ANN. Thus, the first layer of the ANN has allowed the recognition of 66% of nuclear, cytoplasmic and background portions, on average and has also demonstrated very good separation. The second layer of the ANN is able to correctly classify 80% of the cancerous images. This result revealed a good prediction with an accuracy of>65%

REFERENCES

- *The Image Processing Handbook* by John C. Russ, ISBN 0849372542 (2006)
- *Fundamentals of Image Processing* by Ian T. Young, Jan J. Gerbrands, Lucas J. Van Vliet, Paperback, ISBN 90-75691-01-7 (1995)
- *Image Analysis and Mathematical Morphology* by Jean Serra, ISBN 0126372403 (1982)
- *Image Analysis and Mathematical Morphology, Volume 2: Theoretical Advances* by Jean Serra, ISBN 0-12-637241-1 (1988)
- *Scale-Space Theory in Computer Vision* by Tony Lindeberg, ISBN 0-7923-9418-6 (1994)
- *Front-End Vision and Multi-Scale Image Analysis* by Bart M. ter Haar Romeny, Paperback, ISBN 1-4020-1507-0 (2003)
- *Geometry-Driven Diffusion in Computer Vision* by Bart M. ter Haar Romeny (Ed.), ISBN 0792330870{ (1994)
- *Digital Image Processing* by Rafael C. Gonzalez, Richard E. Woods, ISBN 0-201-50803-6 (1992)
- *Digital Image Processing* by William K. Pratt, Paperback, ISBN 0-471-01888-0 (1978)

APPENDIX-I

R, G, B Feature extraction using MATLAB.

```
names = ['u1inter';
'u2super';
'u3super';
'u4inter';
'u5super';
'u6super';
'u7super';
'u8parab';
'u8super';
'u9su+pa';
'u10supe';
'u10su+p';
'u12inte';
'u13supe';
'u14inte';
'u15inte';
'u16inte';
'u17inte';
'u18supe';
'u19inte';
'u20para';
'u21inte';
'u22supe';
'u23inte';
'u24inte';
'u25inte';
```

'u26inte';
'u27supe';
'u28inte';
'u29supe';
'u30inte';
'u31supe';
'u32inte';
'u33inte';
'u34inte';
'u35inte';
'u36supe';
'u38inte';
'u39inte';
'u40supe';
'u41supe';
'u42supe';
'u43inte';
'u44supe';
'u45supe';
'u46supe';
'u47supe';
'u48para';
'u49para';
'u50inte';
'u51inte';
'u52para';
'u53supe';
'u54para';
'u55doub';
'u56para';
'u57mali';

'u58mali';
'u60mali';
'u61mali';
'u62mali';
'u63mali';
'u64mali';
'u65mali';
'u66mali';
'u67mali';
'u68mali';
'u69mali';
'u70mali';
'u71mali';
'u73mali';
'u74mali';
'u75mali';
'u76mali';
'u77mali';
'u78mali';
'u79mali';
'u80mali';
'u81mali';
'u82mali';
'u83mali';
'u84mali';
'u85mali';
'u86mali';
'u87mali';
'u88mali';
'u89mali';
'u91mali';

```
'u92mali'
];

for i=1:89
   image_name=strcat(names(i,:),'.jpg');
   image_c=strcat(names(i,:),'_c.txt_9.txt');
   image_b=strcat(names(i,:),'_b.txt_9.txt');
   image_n=strcat(names(i,:),'_n.txt_9.txt');
a = imread(image_name);
[m,n,p] = size(a);

a = double(a);
for j=1:3
   if(j==1)

        infile=image_c;

   end
   if(j==2)

        infile=image_b;
   end
   if(j==3)

        infile=image_n;
   end


 b = load(infile);
 [k,l] = size(b);
 outfile=strcat(infile,'_gscale.txt');
```

```
fid = fopen(outfile,'w');
%fprintf(fid,'%d %d\n\n',m,n);
for i=1:k
   y=b(i,1);
   x=b(i,2);
      %fprintf(fid,'%d %d              ',x,y);

      if(x<m & y<n & x>=0 & y>=0)
         if((a(x+1,y+1,1)>0))
            fprintf(fid,'%f\n',((a(x+1,y+1,1)+a(x+1,y+1,2)+a(x+1,y+1,3))/3));
         end

      end

      fclose(fid);
      fid = fopen(outfile,'a');


end
end
end

clear all;

%fclose(fid);
close all;


   %%%%Calculate the features at the point (x,y)
      %fid = fopen('feature_test2_c.txt','a');
%close all;
```

APPENDIX-II

Statistical features as an input for ANN.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<fstream.h>
#include<string.h>
float data[3][3];
void calculate(FILE*);

void main()
{
char names[89][20]={"u1inter",
"u2super",
"u3super",
"u4inter",
"u5super",
"u6super",
"u7super",
"u8parab",
"u8super",
"u9su+pa",
"u10supe",
"u10su+p",
"u12inte",
"u13supe",
"u14inte",
"u15inte",
"u16inte",
"u17inte",
"u18supe",
"u19inte",
"u20para",
"u21inte",
"u22supe",
"u23inte",
"u24inte",
"u25inte",
"u26inte",
"u27supe",
"u28inte",
```

"u29supe",
"u30inte",
"u31supe",
"u32inte",
"u33inte",
"u34inte",
"u35inte",
"u36supe",
"u38inte",
"u39inte",
"u40supe",
"u41supe",
"u42supe",
"u43inte",
"u44supe",
"u45supe",
"u46supe",
"u47supe",
"u48para",
"u49para",
"u50inte",
"u51inte",
"u52para",
"u53supe",
"u54para",
"u55doub",
"u56para",
"u57mali",
"u58mali",
"u60mali",
"u61mali",
"u62mali",
"u63mali",
"u64mali",
"u65mali",
"u66mali",
"u67mali",
"u68mali",
"u69mali",
"u70mali",
"u71mali",
"u73mali",
"u74mali",
"u75mali",
"u76mali",
"u77mali",

```
"u78mali",
"u79mali",
"u80mali",
"u81mali",
"u82mali",
"u83mali",
"u84mali",
"u85mali",
"u86mali",
"u87mali",
"u88mali",
"u89mali",
"u91mali",
"u92mali"
};
char*image_rgb,*infile;
char temp[30]={""};
char temp1[30]={""};

FILE*fp,*fp1,*fp2,*fp3;
float val;
int count=0;

image_rgb=(char*)malloc(50*sizeof(char));
infile=(char*)malloc(50*sizeof(char));
for(int i=0;i<89;i++)
{
        //printf("image %d",i);
        //strcpy(temp,names[i]);
        for(int j=0;j<3;j++)
        {

                strcpy(temp,names[i]);
                strcpy(temp1,names[i]);
                //printf("%s\t%s\n",infile,image_rgb);
                if(j==0)
                {
                        //printf("hiadsasssdsa\n");
                        infile=strcat(temp,"_b.txt_9.txt_gscale.txt");
                        image_rgb=strcat(temp1,"_b.txt_feat.txt");
                        printf("%s\t%s\n",infile,image_rgb);
                }
                else if(j==1)
                {
                        infile=strcat(temp,"_c.txt_9.txt_gscale.txt");
                        image_rgb=strcat(temp1,"_c.txt_feat.txt");
```

```
        }
        else
        {
                infile=strcat(temp,"_n.txt_9.txt_gscale.txt");
                image_rgb=strcat(temp1,"_n.txt_feat.txt");
        }

//printf("hiadsasssdsa\n");
fp=fopen(infile,"r");
fp1=fopen("train_final.txt","a");
fp2=fopen(image_rgb,"r");
fp3=fopen("train_output.txt","a");
if(fp==NULL||fp1==NULL||fp2==NULL||fp3==NULL)
{
        printf("Unable to open file\n");
        exit(0);
}
//printf("hiadsasssdsa\n");
fprintf(fp1,"\n\n");
count=0;
while((!feof(fp))||(!feof(fp2)))
{
        count++;
        //printf("%d\n",count);
        //if(count>=20)
        //{
        //      continue;
        //}
        //else
        //{
        for(int i=0;i<3;i++)
        {
                fscanf(fp2,"%f",&val);
                fprintf(fp1,"%f ",val/255);

                for(int j=0;j<3;j++)
                {
                        fscanf(fp,"%f",&data[i][j]);
                        data[i][j]=data[i][j]/255;
                        //printf("%d ",data[i][j]);
                }
                //printf("\n");
        }
        if(j==0)
        {
        fprintf(fp3,"0 0 1\n");
```

```
                }
                else if(j==1)
                {
                fprintf(fp3,"0 1 0\n");
                }
                else
                {
                fprintf(fp3,"1 0 0\n");
                }
                calculate(fp1);
                //}
        }


        fclose(fp3);
        fclose(fp1);
        fclose(fp);
        fclose(fp2);
        }
}
}

void calculate(FILE* fp1)
{
        float dx,dy;
        float max=0.0;
        float min=0.0;
        float avg=0.0;
        float edge_magnitude=0.0;
        long double edge_direction=0.0;
        float variance=0.0;
        float std_dev=0.0;
        float area_desc=0.0;
        float mean_energy=0.0;
        float energy_var=0.0;
        float entropy=0.0;
        float skewness=0.0;
        float kurtosis=0.0;
        float compactness=0.0;
        float difference=0.0;
        float contrast=0.0;
        float avg_bg=0.0;


        dx=(float)(data[0][2]+(2*data[1][2])+data[2][2])-(data[0][0]+(2*data[1][0])+data[2][0]);
```

```
dy=(float)(data[2][0]+(2*data[2][1])+data[2][2])-(data[0][0]+(2*data[0][1])+data[0][2]);


edge_magnitude=sqrt(pow(dx,2)+pow(dy,2));  //edge magnitude
//printf("edge===%f",edge_magnitude);
edge_direction=atanl(dy/dx);          //edge direction
fprintf(fp1,"%f %lf ",edge_magnitude,edge_direction);



//printf("%d %d\n",edge_magnitude,edge_direction);
//max,min,avg
max=min=(data[0][0]);
for(int i=0;i<3;i++)
{
        for(int j=0;j<3;j++)
        {
                avg+=(data[i][j]);
                if(max<(data[i][j]))
                        max=(data[i][j]);
                if(min>(data[i][j]))
                        min=(data[i][j]);
        }
}
//printf("%f",avg);
avg=(float)avg/9;
printf("%f\n",avg);
//getch();
fprintf(fp1,"%f %f %f ",min,max,avg);

//variance,mean energy,entropy,std_dev,area_desc,skewness,curtosis,difference,contrast
for(i=0;i<3;i++)
{
        for(int j=0;j<3;j++)
        {
                variance+=pow((data[i][j]-avg),2);
                //printf("entropyyyyyyyyy ========%f,%f\n",data[i][j],avg);
                mean_energy+=pow(data[i][j],2);
                entropy+=data[i][j]/(avg*9);
                skewness+=pow(data[i][j],3);
                kurtosis+=pow(data[i][j],4);
                if(i==0||i==2||j==0||j==2)
                {
                        avg_bg+=data[i][j];
                }
        }
```

```
        }
        variance=variance/9;
        mean_energy=mean_energy/9;

        entropy=entropy*(1-entropy);
        std_dev=sqrt(variance);
        area_desc=std_dev/avg;
        skewness=(1/pow(std_dev,3))*skewness;
        kurtosis=(1/pow(std_dev,4))*kurtosis;
        difference=avg-avg_bg;
        contrast=difference/(avg+avg_bg);
        fprintf(fp1,"%f %f %f %f %f %f
",variance,mean_energy,std_dev,area_desc,difference,contrast);


        //energy variance
        for(i=0;i<3;i++)
        {
                for(int j=0;j<3;j++)
                {
                        energy_var+=pow((pow(data[i][j],2)-mean_energy),2);
                }
        }
        energy_var=energy_var/9;
        fprintf(fp1,"%f\n",energy_var);
}
```

APPENDIX-III

Artificial Neural Network:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>


void init(int,int,int);
void train(FILE*,FILE*,int,int,int,int);
void parse(struct node*,int,int,int,int);
void sig(struct node*,int);
void bprop(double*,int,int,int);
void clear(int,int,int);
void optimise(int,int,int);
int max(float*,int);


struct node
{
        long double in,out;
};
struct node *innode;
struct node *hnode;
struct node *onode;


struct weight
{
        long double* w;
}warr[2];

void clear_weights(struct weight*,int,int,int);

int f=0;

void main()
{
        int i,h,o,u,y,call=0,flag=1;
        FILE*fp,*fp1,*fp2,*fp3;
        char ch,*wihfile,*whofile;
```

```
        wihfile=(char*)malloc(30*sizeof(char));
        whofile=(char*)malloc(30*sizeof(char));
        srand(time(NULL));
        printf("ENTER THE NUMBER OF INPUT NODES:\n");
        scanf("%d",&i);
        fflush(stdin);
        printf("DO YOU WANT THE PROGRAM TO OPTIMISE THE NUMBER OF
HIDDEN NODES(Y/N)?\n");
        scanf("%c",&ch);
        printf("%c\n",ch);
        fflush(stdin);
        if(ch=='y'||ch=='Y')
        {
                printf("ENTER THE NUMBER TO WHICH YOU WANT THE PROGRAM TO
TEST FOR HIDDEN NODES\n");
                scanf("%d",&y);

        }
        else
        {
                printf("ENTER THE NUMBER OF HIDDEN NODES\n");
                scanf("%d",&y);
                flag=y;
        }
        printf("ENTER THE NUMBER OF OUTPUT NODES\n");
        scanf("%d",&o);

        printf("ENTER THE NUMBER OF ITERATIONS U WANT TO CONTINUE:\n");
        scanf("%d",&u);
        printf("%d       %d\n",flag,y);
        getch();
        for(h=flag;h<=y;h+=2)
        {
                printf("%d\n",h);
        init(i,h,o);
        sprintf(wihfile,"weightih_%d.txt",h);
        sprintf(whofile,"weightho_%d.txt",h);
        fp2=fopen(wihfile,"w");
        fp3=fopen(whofile,"w");
        for(int g=0;g<u;g++)
        {

        fp=fopen("train_final_normalise.txt","r");
        fp1=fopen("train_output.txt","r");

        train(fp,fp1,i,h,o,1);
```

```
}
fp=fopen("train_final_normalise.txt","r");
fp1=fopen("train_output.txt","r");
train(fp,fp1,i,h,o,0);
fclose(fp);
fclose(fp1);


printf("\n\nweights between input and hidden \n");

for(int l=0;l<i*h;l++)
{
        printf("%lf ",warr[0].w[l]);
        fprintf(fp2,"%lf ",warr[0].w[l]);
        if((l+1)%h==0)
        {
                printf("\n");
                fprintf(fp2,"\n");
        }
}



printf("\n\nweights between hidden and output \n");


for(l=0;l<h*o;l++)
{
        printf("%lf ",warr[1].w[l]);
        fprintf(fp3,"%lf ",warr[1].w[l]);
        if((l+1)%o==0)
        {
                printf("\n");
                fprintf(fp3,"\n");
        }
}

}
fclose(fp2);
fclose(fp3);
if(ch=='y'||ch=='Y')
{
        optimise(i,h,o);
}
}
```

```
void init(int i,int h,int o)
{
        long double r;

        innode = (struct node*) calloc(i,sizeof(struct node));
        hnode = (struct node*) calloc(h,sizeof(struct node));
        onode = (struct node*) calloc(o,sizeof(struct node));
        warr[0].w=(long double*)calloc(i*h,sizeof(long double));
        warr[1].w=(long double*)calloc(h*o,sizeof(long double));
        printf("im allocating....\n");
        //getch();
        for(int m=0;m<2;m++)
        {
                if(m==0)
                {
                        for(int k=0;k<i*h;k++)
                        {
                                r=rand()%10;
                                r=(r-5)/10;
                                if(r)
                                {
                                        warr[m].w[k]=r;
                                }
                                else
                                {
                                        while(!r)
                                        {
                                                r=rand()%10;
                                                r=(r-5)/10;
                                        }
                                        warr[m].w[k]=r;
                                }

                        }
                }
                else
                {
                        for(int k=0;k<h*o;k++)
                        {
                                r=rand()%10;
                                r=(r-5)/10;
                                if(r)
                                {
                                        warr[m].w[k]=r;
```

```
                        }
                        else
                        {
                                while(!r)
                                {
                                        r=rand()%10;
                                        r=(r-5)/10;
                                }
                                warr[m].w[k]=r;
                        }

                }
        }
}
}

// Training the neural net


void train(FILE*fp,FILE*fp1,int i,int h,int o,int call)
{
        int l=0,a=0;//count
        int f=1;
        static int g=0;
        double *arr;


        arr=(double*)calloc(o,sizeof(double));

        if(fp==NULL)
                printf("Empty file handle\n");

        while(!feof(fp) && !feof(fp1))
        {
                for(int p=0;p<i;p++)
                {

                        fscanf(fp,"%lf",&innode[p].in);
                innode[p].out=innode[p].in;
                }

                for(int a=0;a<o;a++)
                {

                        fscanf(fp1,"%lf",&arr[a]);
```

```
                }

                        parse(innode,i,h,o,call);


                        if(call)
                        {
                                bprop(arr,i,h,o);
                        }

                        a=0;
                        l=0;
                        continue;

                l++;
                f++;
                a++;
        }
        free(arr);
        fclose(fp);
        fclose(fp1);
}

//Parse each set of values from input file

void parse(struct node *innode,int i,int h,int o,int call)

{

        int x=0;
        FILE* fp1;
        static int qw=0;
        char* file_name;
        file_name=(char*)malloc(40*sizeof(char));
        clear(i,h,o);

        for(int q=0;q<h;q++)
        {
                for(int g=0;g<i;g++)
                {
                        hnode[q].in+=innode[g].out*warr[0].w[x];
                        x++;
                }
        }

        sig(hnode,h);
```

```
x=0;
for(int g=0;g<o;g++)
{
        for(int p=0;p<h;p++)
        {
                onode[g].in+=hnode[p].out*warr[1].w[x];
                x++;
        }
}
sig(onode,o);

if(call==0)
{



        qw++;
        f++;

        sprintf(file_name,"output_hidden_%d.txt",h);
        fp1=fopen(file_name,"a");
        if(fp1==NULL)
        {

                printf("%d    file cant b opened\n",qw);
                getch();

        }
                for(int d=0;d<o;d++)
                {
                        fprintf(fp1,"%lf\t",onode[d].out);

                }
                        fprintf(fp1,"\n");

                        fclose(fp1);
 }
 free(file_name);


}


void sig(struct node*x,int size)
{
```

```
        for(int i=0;i<size;i++)
        {
        if((1/(1+exp(-x[i].in))))
                x[i].out=(1/(1+exp(-x[i].in)));
        }
}

void bprop(double* arr,int i,int h,int o)
{
        long double* error,*error2;
        long double x=0.0;
        int y=0;
        int k=0;
        error=(long double*)calloc(o,sizeof(long double));
        for(int l=0;l<o;l++)
        {
                error[l]=0.1*onode[l].out*(1-onode[l].out)*(arr[l]-onode[l].out);
        }
        for(int f=0;f<h*o;f++)
        {
                k=0;
                while(k<o)
                {
                        warr[1].w[f]=warr[1].w[f]+(error[k]*hnode[(int)f/o].out);
                        k++;
                        f++;
                }
                f--;
        }
        error2=(long double*)calloc(h,sizeof(long double));
        for(l=0;l<h;l++)
        {
                for(int m=0;m<o;m++)
                {
                        x+=error[m]*warr[0].w[y];
                        y++;
                }
                error2[l]=0.1*hnode[l].out*(1-hnode[l].out)*x;
        }
        for(f=0;f<(i*h);f++)
        {
                k=0;
                while(k<h)
                {
                                warr[0].w[f]=warr[0].w[f]+(error2[k]*innode[(int)f/h].out);
```

```
                        k++;
                        f++;
                }
                f--;
        }


free(error);
free(error2);
}


void clear(int i,int h,int o)
{
        for(int p=0;p<h;p++)
        {
                hnode[p].in=hnode[p].out=0.0;
        }
        for(int k=0;k<o;k++)
        {
                onode[k].in=onode[k].out=0.0;
        }
}


void optimise(int a,int b,int c)
{
        FILE*fp,*fp1,*results;
        int max1=0,max2=0,hidden_optimised=0;
        char*file_name,*file_results;
        float temp=0.0;
        float
*temp1,*temp2,*acc_count,**accuracy,**ppv,**sensitivity,**specificity,**npv,**mcc;
        float *tp,*falsep,*fn,*tn;
        acc_count=(float*)malloc(b*sizeof(float));
        file_name=(char*)malloc(30*sizeof(char));
        temp1=(float*)malloc(c*sizeof(float));
        temp2=(float*)malloc(c*sizeof(float));
        accuracy=(float**)malloc(b*sizeof(float*));
        ppv=(float**)malloc(b*sizeof(float*));
        specificity=(float**)malloc(b*sizeof(float*));
        sensitivity=(float**)malloc(b*sizeof(float*));
        npv=(float**)malloc(b*sizeof(float*));
```

```
mcc=(float**)malloc(b*sizeof(float*));
tp=(float*)malloc(c*sizeof(float));
falsep=(float*)malloc(c*sizeof(float));
fn=(float*)malloc(c*sizeof(float));
tn=(float*)malloc(c*sizeof(float));



for(int q=0;q<b;q++)
{
        accuracy[q]=(float*)malloc(c*sizeof(float));
        ppv[q]=(float*)malloc(c*sizeof(float));
        specificity[q]=(float*)malloc(c*sizeof(float));
        sensitivity[q]=(float*)malloc(c*sizeof(float));
        npv[q]=(float*)malloc(c*sizeof(float));
        mcc[q]=(float*)malloc(c*sizeof(float));

}

file_results=(char*)malloc(30*sizeof(char));

for(int t=0;t<b;t++)
{
        acc_count[t]=0.0;
}
results=fopen("results_2.txt","w");
for(int i=1,d=0;i<b;i+=2,d++)
{
        sprintf(file_name,"output_hidden_%d.txt",i);
        fp=fopen(file_name,"r");
        puts(file_name);
        fp1=fopen("train_output.txt","r");
        if(fp1==NULL)
        {
                printf("Unable to open train_output file\n");
                getch();
        }
        if(fp==NULL)
        {
                printf("Unable to open file %s\n",file_name);
                getch();
        }
        else
        {
```

```
while(!feof(fp1)&&!feof(fp))
{
        for(int x=0;x<c;x++)
        {

                fscanf(fp,"%f",&temp1[x]);
                fscanf(fp1,"%f",&temp2[x]);
        }
        max1=max(temp1,c);
        max2=max(temp2,c);
        for(int g=0;g<c;g++)
        {
                if(max1==max2)
                {
                        printf("%d   %d\n",max1,max2);
                        acc_count[d]+=1.0;
                        if(max1==g)
                        {
                                tp[g]+=1.0;
                                printf("heree....%f",tp[g]);
                                getch();
                        }
                        else
                        {
                                tn[g]+=1.0;
                        }
                }

                else
                {
                        if(max1==g)
                        {
                                falsep[g]+=1.0;
                        }
                        else
                        {
                                fn[g]+=1.0;
                        }
                }

        }
}
for(int r=0;r<c;r++)
{
        accuracy[d][r]=(tp[r]+tn[r])/(tp[r]+falsep[r]+fn[r]+tn[r]);
        ppv[d][r]=(tp[r])/(tp[r]+falsep[r]);
```

```
                                        sensitivity[d][r]=(tp[r])/(tp[r]+fn[r]);
                                        specificity[d][r]=(tn[r])/(falsep[r]+tn[r]);
                                        npv[d][r]=(tn[r])/(tn[r]+fn[r]);
                                        mcc[d][r]=((tp[r]*falsep[r])-
(tn[r]*fn[r]))/sqrt((tp[r]+tn[r])*(tp[r]+fn[r])*(fn[r]+tn[r])*(fn[r]+falsep[r]));
                                }
                        }
                        fclose(fp);
                }
                for(int u=0;u<d;u++)
                {
                        for(int cn=0;cn<c;cn++)
                        {
                                fprintf(results,"accuracy count for %d class when %d hidden
nodes%f\n",cn,(u*2+1),accuracy[u]);
                                fprintf(results,"ppv[%d][%d]====%f\n",(u*2+1),cn,ppv[u]);

                fprintf(results,"sensitivity[%d][%d]=====%f\n",(u*2+1),cn,sensitivity[u]);

                fprintf(results,"specificity[%d][%d]=====%f\n",(u*2+1),cn,specificity[u]);
                                fprintf(results,"npv[%d][%d]====%f\n",(u*2+1),cn,npv[u]);
                                fprintf(results,"mcc[%d][%d]=====%f\n\n\n\n",(u*2+1),cn,mcc[u]);
                        }
                        fprintf(results,"\n\n\n\n\n");
                }
                        hidden_optimised=temp*2+1;
                printf("THE NETWORK IS FOUND TO BE OPTIMISED AT %d NUMBER OF
HIDDEN NODES\n",hidden_optimised);
                printf("SO PLEASE NOTE DOWN THE NUMBER OPTIMISED NUMBER OF
HIDDEN NODES FOR FURTHER USE\n");
}

int max(float *arr,int size)
{

                float max=arr[0];
                int ret_value=0;
                for(int a=0;a<size;a++)
                {
                        if(arr[a]>max)
                        {
                                max=arr[a];
                                ret_value=a;
                        }
                }
                return ret_value;
```

```
}

void clear_weights(struct weight* warr,int i,int h,int o)
{

        for(int m=0;m<2;m++)
        {
                if(m==0)
                {
                        for(int k=0;k<i*h;k++)
                        {
                                warr[m].w[k]=0.0;



                        }
                }
                else
                {
                        for(int k=0;k<h*o;k++)
                        {
                                warr[m].w[k]=0.0;


                        }
                }
        }
}
```

APPENDIX IV

R,G,B Features of all the available images

```
#include<stdio.h>
#include<malloc.h>
#include<string.h>
#include<math.h>
#include<conio.h>

float data[3][3];

void calculate(FILE*);
void main()
{
        char names[89][20]={"u1inter",
"u2super",
"u3super",
"u4inter",
"u5super",
"u6super",
"u7super",
"u8parab",
"u8super",
"u9su+pa",
"u10supe",
"u10su+p",
"u12inte",
"u13supe",
"u14inte",
"u15inte",
"u16inte",
"u17inte",
"u18supe",
"u19inte",
"u20para",
"u21inte",
"u22supe",
"u23inte",
"u24inte",
"u25inte",
"u26inte",
"u27supe",
"u28inte",
```

"u29supe",
"u30inte",
"u31supe",
"u32inte",
"u33inte",
"u34inte",
"u35inte",
"u36supe",
"u38inte",
"u39inte",
"u40supe",
"u41supe",
"u42supe",
"u43inte",
"u44supe",
"u45supe",
"u46supe",
"u47supe",
"u48para",
"u49para",
"u50inte",
"u51inte",
"u52para",
"u53supe",
"u54para",
"u55doub",
"u56para",
"u57mali",
"u58mali",
"u60mali",
"u61mali",
"u62mali",
"u63mali",
"u64mali",
"u65mali",
"u66mali",
"u67mali",
"u68mali",
"u69mali",
"u70mali",
"u71mali",
"u73mali",
"u74mali",
"u75mali",
"u76mali",
"u77mali",

```
"u78mali",
"u79mali",
"u80mali",
"u81mali",
"u82mali",
"u83mali",
"u84mali",
"u85mali",
"u86mali",
"u87mali",
"u88mali",
"u89mali",
"u91mali",
"u92mali"
};

        FILE*fp,*fp1,*fp2;
        int row,col,check=1;
        char c,*temp1;
        temp1=(char*)malloc(30*sizeof(char));

        for(int sd=0;sd<89;sd++)
        {

                strcpy(temp1,names[sd]);
                puts(temp1);
                strcat(temp1,"_rgb.txt");
                fp1=fopen(temp1,"r");

                if(fp1==NULL)
                {
                        printf("Unable to open file\n");
                }
                strcpy(temp1,names[sd]);
                strcat(temp1,"_15_features.txt");
                fp2=fopen(temp1,"w");
                if(fp2==NULL)
                {
                        printf("Unable to open write file\n");
                }
                fscanf(fp1,"%d",&row);
                printf("%d\n",row);
                fscanf(fp1,"%d",&col);
                printf("%d",col);
        float **image,***image_rgb,temp=0.0;
        image=(float**)malloc(row*sizeof(float*));
```

```
temp1=(char*)malloc(50*sizeof(char));

for(int sg=0;sg<row;sg++)
        image[sg]=(float*)malloc(col*sizeof(float));


image_rgb=(float***)malloc(row*sizeof(float**));
for(int v=0;v<row;v++)
{
        image_rgb[v]=(float**)malloc(col*sizeof(float*));
}

for(int sr=0;sr<row;sr++)
{
        for(int sp=0;sp<col;sp++)
        {
                image_rgb[sr][sp]=(float*)malloc(3*sizeof(float));
        }
}



        printf("\nPlease  be patient while the parameters are being calculated....");

        for(int b=0;b<row&&!feof(fp1);b++)
        {
                for(int c=0;c<col&&!feof(fp1);c++)
                {
                        fscanf(fp1,"%f %f
%f\n",&image_rgb[b][c][0],&image_rgb[b][c][1],&image_rgb[b][c][2]);


        image[b][c]=(((image_rgb[b][c][0]/255)+(image_rgb[b][c][1]/255)+(image_rgb[b][c][2]/
255))/3);
                }
        }

        for(int i=0;i<row;i++)
        {
                for(int j=0;j<col;j++)
                {
                        if(i>=row||j>=col)
                        {
                                break;
                        }
```

```
                        for(int l=i-1;l<=i+1;l++)
                        {
                                for(int m=j-1;m<=j+1;m++)
                                {
                                        if(l<0 || m<0 || l>=row || m>=col)
                                        {
                                                data[l-(i-1)][m-(j-1)]=0.0;
                                        }
                                        else
                                        {
                                                if(l==i&&m==j)
                                                        fprintf(fp2,"%f %f
%f",image_rgb[l][m][0]/255,image_rgb[l][m][1]/255,image_rgb[l][m][2]/255);
                                                data[l-(i-1)][m-(j-1)]=image[l][m];
                                        }
                                }
                        }

                        calculate(fp2);
                }
        }

        fclose(fp1);
        fclose(fp2);
        }

}


void calculate(FILE* fp1)
{
        float dx,dy;
        float max=0.0;
        float min=0.0;
        float avg=0.0;
        float edge_magnitude=0.0;
        long double edge_direction=0.0;
        float variance=0.0;
        float std_dev=0.0;
        float area_desc=0.0;
        float mean_energy=0.0;
        float energy_var=0.0;
        float entropy=0.0;
        float skewness=0.0;
        float kurtosis=0.0;
```

```
float compactness=0.0;
float difference=0.0;
float contrast=0.0;
float avg_bg=0.0;


dx=(float)(data[0][2]+(2*data[1][2])+data[2][2])-(data[0][0]+(2*data[1][0])+data[2][0]);
dy=(float)(data[2][0]+(2*data[2][1])+data[2][2])-(data[0][0]+(2*data[0][1])+data[0][2]);


edge_magnitude=sqrt(pow(dx,2)+pow(dy,2));        if(dx)
        edge_direction=atanl(dy/dx);
else                                                    edge_direction=0.0;
fprintf(fp1," %f %lf ",edge_magnitude,edge_direction);



max=min=(data[0][0]);
for(int i=0;i<3;i++)
{
        for(int j=0;j<3;j++)
        {

                avg+=(data[i][j]);
                if(max<(data[i][j]))
                        max=(data[i][j]);
                if(min>(data[i][j]))
                        min=(data[i][j]);
        }
}
        avg=(float)avg/9;

fprintf(fp1,"%f %f %f ",min,max,avg);

        for(i=0;i<3;i++)
{
        for(int j=0;j<3;j++)
        {
                variance+=pow((data[i][j]-avg),2);
                        mean_energy+=pow(data[i][j],2);
                if(avg)
                        entropy+=data[i][j]/(avg*9);
                else
                        entropy=0.0;
                skewness+=pow(data[i][j],3);
                kurtosis+=pow(data[i][j],4);
```

```
                    if(i==0||i==2||j==0||j==2)
                    {
                            avg_bg+=data[i][j];
                    }
            }
        }
        variance=variance/9;
        mean_energy=mean_energy/9;

        entropy=entropy*(1-entropy);
        std_dev=sqrt(variance);
        area_desc=std_dev/avg;
        skewness=(1/pow(std_dev,3))*skewness;
        kurtosis=(1/pow(std_dev,4))*kurtosis;
        difference=avg-avg_bg;
        contrast=difference/(avg+avg_bg);
        fprintf(fp1,"%f %f %f %f %f %f
",variance,mean_energy,std_dev,area_desc,difference,contrast);


        //energy variance
        for(i=0;i<3;i++)
        {
                for(int j=0;j<3;j++)
                {
                        energy_var+=pow((pow(data[i][j],2)-mean_energy),2);
                }
        }
        energy_var=energy_var/9;
        fprintf(fp1,"%f\n",energy_var);
}
```