DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF ENGINEERING,

THE UNIVERSITY OF SHEFFIELD

COM 6906

Machine Learning System to recognise cancer from a gene-expression profile

Direnc Pekaslan

Supervisors: Dr. Dawn Walker Dr.Daniele Tartarini

This report is submitted in partial fulfilment of the requirement

for the degree of MSc in Advanced Computer Science by Direnc Pekaslan

September 2015

Signed Declaration

All sentences or passages quoted in this dissertation from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this dissertation have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amount to plagiarism and will be considered grounds for failure in this dissertation and the degree examination as a whole.

Name: Direnc Pekaslan

Signature: ___________________

Date: September 2015.

Abstract

Cancer is the leading cause of death by disease in the world. In 2012, there were 32.6 million people (within five years of diagnosis) who were suffering from cancerous diseases and 8.2 million of these resulted in death (www1, 2015; www2, 2015). Due to the unique response of each patient to treatment, clinicians need accurate information of diagnosis and prognosis in order to be able to tailor treatment successfully. The purpose of this project is to develop an accurate computational tool which can predict information such as the stage, metastasis capability and/or typology of cancer from a publicly available gene-expression profile based on machine learning techniques. In this report, relevant literatures that have used a multilayer neural network in gene expression datasets to classify and predict survivability and identify biomarkers of cancer are investigated. A summary of the main findings suggests that a multilayer neural network is capable of accurate classification and prediction in cancer gene expression profiles.

ACKNOWLEGEMENTS

I would like to express my gratitude to my supervisors, Dr Dawn Walker and Dr Daniele Tartarini, for their generous help and precious guidance without which this project would not have been possible.

I also express my appreciation to my family for their unconditional support throughout my life.

Contents

Glossary:

DNA: Base hereditary biological instruction for all eukaryotic cells which are composed of nucleotides (WWW8, 2015)
RNA: Ribonucleic acid carries transmits genetic information from DNA to proteins.
Nucleotides: Structural components of DNA and RNA.
mRNA: Messenger RNA is a molecule that is complementary to one of the DNA strands of a gene.
cDNA: Complementary DNA is synthesized molecule from mRNA.
Probe: A probe is a single-stranded sequence of DNA or RNA (WWW12, 2015)
Gene expression: A process of encoding new functional product from a gene to be assembled to a protein molecule (WWW12 , 2015).

List of Figures

List of Tables

Chapter 1

Introduction

Cancer is the leading cause of death by disease in the world. In 2012, there were 32.6 million people (within five years of diagnosis) who were suffering from cancerous diseases and 8.2 million of these resulted in death (www1, 2015; www2, 2015).

Today, cancer tumours can be categorised into 100 different types and they are usually named by the name of the organ in which they appear. Also, those named tumours are classified into different subtypes based on similarities in their genetic characteristics. Although cancer subtypes can be the same for different people and the treatment of them can be classified as chemotherapy, radiotherapy, radiation, hormonal therapy and so on, each patient's response is unique to the administered treatments (www3, 2015). This suggests that tailoring treatment individually has a significant role in the cure of cancer; however, tailoring treatment requires accurate information on the cancer subtypes in the early stages. Previous studies have shown that the technique called gene expression profiling has the capability (Brown & Botstein, 1999) to understand cancer subgroups, metastasis (which is the spreading capability of a tumour) or the recurrence possibility of cancer by using computer simulations and the present study seeks to develop a computational tool for investigating gene expression profiling to give accurate information to clinicians.

1.1 Goals

The main aim of this project is to develop a computational tool that can be trained by publicly available gene expression profiles and then the trained tool will be able to predict information like the stage, metastasis capability and/or typology of cancer accurately with given gene expression profiles.

The second part of the project will be analysing performance of serial tool code and profiling investigations to make software faster to implement it on GPU in the future

1.2 Objectives

As objectives of project, it is intended to achieve that;

- Developing an open-source software library, named CUNERNET (that is, CUDA Neural Network) using C++11 language.
- Designing a neural network software in serial algorithms which can be trained by any suitable gene expression profiles and can be extended to a multilayer neural network structure.
- Using proper software development techniques such as a concurrent versioning system, continuous integration and unit testing to release the software with BSD3 license and reach the widest possible community.
- Pre-processing/choosing available gene expression datasets for training and testing.
- Training the neural network software by giving the obtained dataset with feedforward/backpropagation algorithms.
- Testing the trained tool by a given gene expression profile for prediction.
- Optimising/re-implementing the written codes using the Google Performance Analysis Tool.
- Identifying/implementing parts of the algorithm on GPU using CUDA.
- Comparing the timing results of GPU and CPU implementations.

1.3 Structure of the dissertation

The rest of the dissertation is structures as follows;

2 Background information and literature review: detailed background information about gene expression profiling and the neural network technique in machine learning.

3 Requirement analysis: analyses the requirements of the project, configuration and environment settings

4 System design : Describes the chosen design techniques along with both the benefits and drawbacks

5 Implementation, testing and evaluation: This chapter will describe neural network training implementation with Cunernet, testing results, and finally an evaluation of work will be explained.

6 Discussion and future works: A critical discussion of findings; the achieved goals are discussed and possible directions for future research are recommended.

Chapter 2

Background and Literature review

This chapter starts by describing the terminology, discussing the theory of relevant topics to the project, and describing previous studies that have been conducted on cancer diagnosis/detection by using the machine learning technique on gene expression profiles.

2.1 Cancer Disease

Our bodies consist of more than a hundred million cells (www13) and each cell contains chromosomes that consist of long strings of DNA (deoxyribonucleic acid) in their nucleus. DNA is a basic hereditary biological instruction for all eukaryotic cells and it is made up thousands of genes that determine cell behaviours, such as death or division, by encoding RNA or producing proteins (www13), which will be explained in Section 2.1.2 in detail. During cell division, those genes may be damaged (gene mutation) and thereafter the damaged gene may stop instructing the cell properly which leads to abnormal cell proliferation (www14) (*see* Figure 2.1).

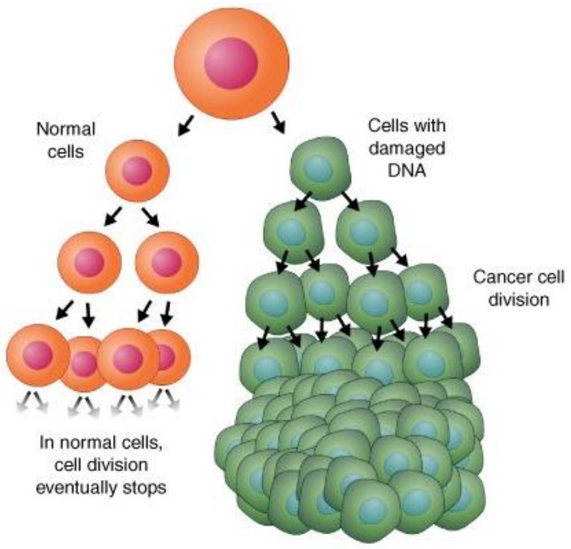


Figure 2.1: Cancer cell proliferation: After a gene damaged, the cell division/die procedure is unbalanced which cause to cell proliferation (www14).

Due to this abnormal proliferation/accumulation, a tumour develops and might spread to other organs through the bloodstream, a process which is known as metastasis (*see* Figure 2.2) which can be called cancer disease.

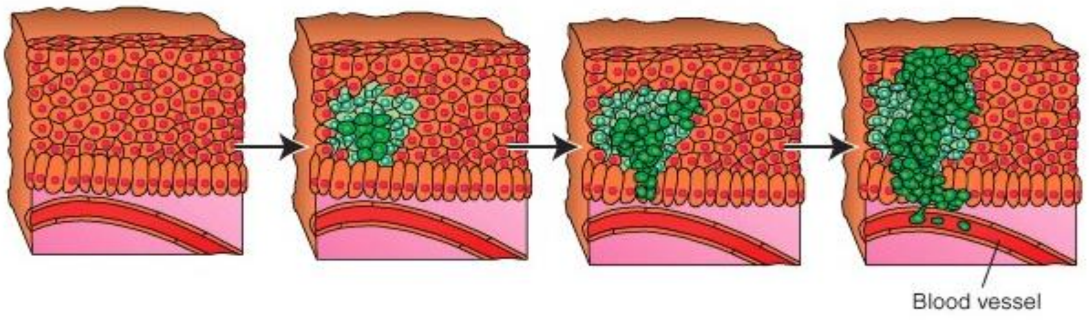


Figure 2.2 Cancer Metastasis;Cell proliferation, tumour developing and metastasis (www14).

2.1.1 Detection/Diagnosing Cancer

Different cancer types can be seen in any part of the human body and it is also possible that different groups of cancer can appear in the same organ (www4, 2015; Curtis *et al*., 2012). Today it is known that there are more than 200 different cancer types (www15, 2015). However recent studies have revealed that there are still many cancer subgroups to be discovered and classified. Curtis *et al*.'s study, which was conducted using gene expression profiles, proved that breast cancer consists of at least ten different diseases (Curtis *et al*., 2012) and another genomic study has shown that breast cancer of type HER2+ can be further classified into four different subtypes (Prat *et al*., 2014). Despite the fact that different types of tumour/cancer may occur in the same organ, each one of those groups responds to drugs in different ways and requires completely different treatments (Garnett *et al*., 2012). As studies have shown, gene expression profiles have a significant ability to distinguish/detect different subgroups of cancer which is crucial and accurate information that can be used to tailor cancer treatments (Geeleher, Cox & Huang, 2014; Sotiriou & Piccart, 2007).

In addition to accurate detection, cancer diagnosis in the early stages also has an important impact on its successful treatment (www5, 2015). Cancer classifications are done by clinicians depending on size of tumour and the quantity of spreading in other organs. Typically there are four levels for staging cancer, ranging from *Stage I* small cancer located in a single organ, to *Stage IV* cancer of substantial size spread in different organs (i.e. metastases) (www6, 2014). According to Cancer Research UK, the chance of lung cancer patients surviving is 70% in the case of diagnosing at stage I over one year. However this percentage drops to less than 15% if it is diagnosed at stage IV (www7, 2014). In addition, the ten-year survival chance for colorectal cancer is 94% if detected at stage I, whereas it is only 4% at stage IV (www7, 2014). Furthermore, Cancer Research UK stated that more than 1200 patients' lives would be saved by a 1% rise in cancers diagnosed at stages I or II in the five years following diagnosis (www7, 2014).

Apart from reducing mortality, early detection has some additional advantages which relate to the cost of treatment. A report which was prepared by the Incisive Health Team for Cancer Research UK showed that total treatment cost per cancer patient can vary enormously depending on which stage it is diagnosed at (*see* Table 2.1) (Incisive Health, 2014).

	Total cost per-patient	Number of patients	Total cost to the NHS
Stage 1	£3,747.63	2,931	£10,988,326.24
Stage 2	£9,810.70	7,237	£71,010,839.78
Stage 3	£13,974.87	7,450	£104,132,075.80
Stage 4	£12,518.58	5,690	£71,235,854.18

Table 2.1: NHS wide cost associated with the colon cancer treatment pathway, including recurrence (Incisive Health,2014)

In the light of these facts, early and accurate cancer diagnosis is vital both for successful treatment and to reduce costs.

Studies have proved that early/accurate diagnosis can only be achievable by investigating molecular level elements rather than waiting for cancer cells to accumulate and spread to surrounding tissues, thereby making a tumour recognisable (www6, 2014). At the time of this research, the number of publications (7510 regarding Machine Learning and Cancer searches in PubMed) provides evidence that gene expression profiling is a highly reliable and promising approach to the accurate prediction of cancer (Ross *et al*., 2000; Kourou *et al*., 2015).

2.1.2 Gene expression profiling and DNA microarray

The code of DNA is made up four chemical bases: adenine (A), guanine (G), cytosine (C) and thymine (T), and by pairing these chemicals, the two-strand helix structure of DNA is built up. As described in section 2.1, DNA contains instructions to develop, survive and reproduce for organisms and these functions can be carried out by producing proteins. The genes, segments of DNA, encode the instructions to create protein molecules (WWW8, 2015). In order to produce proteins, enzyme RNA polymerase encodes DNA. After that, through the encoding process (transcription), pre-mRNA and messenger RNA (mRNA) are produced respectively. The produced mRNA carries some sequences to the ribosome where the actual production of protein will take place. After the mRNA sequences reach the ribosome, a reading process (translation) is initiated by the ribosome. By this reading process, protein molecules can be produced in a wild polypeptide form depending on the sequences of mRNA (WWW8, 2015).

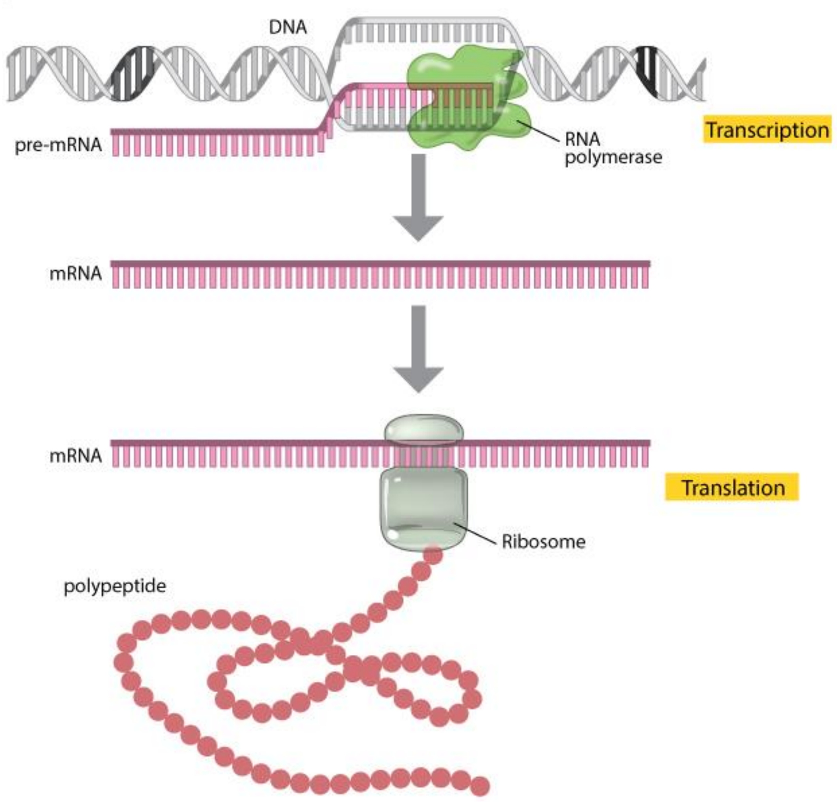


Figure 2.3: Protein synthesis through the process of transcription and transaction (www9, 2015)

The wild polypeptide produced by the ribosome in the translation process folds up spontaneously in its three-dimension shape according to Anfinsen’s dogma (Anfinsen, 1973). The function of the protein is determined by its shape and a misfolded shape leads to a defective protein. The localized accumulation of these misfolded proteins causes some diseases such as neurodegeneration, diabetes and Alzheimer’s. Xu *et al*. (2011) stated that misfolded shape aggregation may also be a reason for cancer diseases (Xu *et al*., 2011).

In biological terms, gene expression can be defined as the process of translation into mRNA or to protein, and this definition can be broadened to include the synthesising of a new functional product from DNA genes.

These manufactured products (proteins) govern the cell functions, and the quantity of a particular protein can reflect the activity/function of the cell. Therefore, determination of these expressed genes gives a picture of cellular function, and this is defined as gene expression profiling (www10, 2014). A technique to quantify thousands of gene expressions simultaneously is called microarray technology. The processes of microarray are implemented as described next.

First, spots of a solid surface are filled by specific DNA sequences, known as probes. Second, mRNA is taken from a body cell (RNA extraction in Figure 2.4) and by reverse transcriptase enzyme it is converted to cDNA (Yang, 2002). Because messenger RNA is produced by transformation from the DNA strand, it gives mRNA the capability of binding to the DNA again (nature.com, 2015). After the reverse transcriptase process, fluorescent nucleotides are attached to the cDNA in order to identify when they are binded to probes on the solid surface (Schena *et al*., 1995). Thereafter, the researcher places the labelled cDNAs into a DNA microarray slide. Some particular cDNAs bind to their complementary part of DNA, which is known as hybridization (Schena *et al*., 1995). By scanning the surface, intense fluorescents are identified and if there is a particularly intense spot on the solid surface, it indicates that the area is producing many molecules of mRNA (Schena *et al*., 1995). By examining the whole surface in terms of the quantity of fluorescents, scientists can gain a more accurate understanding of thousands of gene activities. Consequently the data obtained from gene expression profiling is called gene expression signature, and the gene expression level refers to the amount of detected mRNA in the sample.

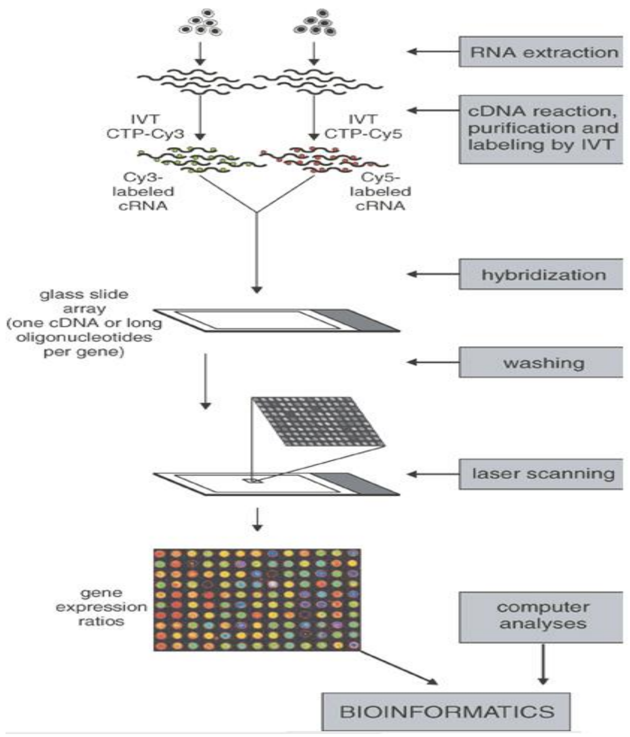


Figure 2.4: Microarray Technique procedures as follow; mRNA are isolated and cDNA is made. Then cDNA is labelled with fluorescent nucleotides thereafter the labelled cDNA is put to glass slide and the slide is scanned by laser followed by computer analysis of the intensity image (www11, 2015).

2.1.3 Genomic dataset

The main three concerns in cancer prediction/prognosis are cancer susceptibility (that is, risk assessment), cancer recurrence and cancer survivability. Naturally these three predictions could be conducted by considering multiple clinical factors such as the age of the patient, the grade and size of the tumour and so on. However these 'macro-scale' clinical factors are generally inadequate for accurate prediction/diagnosis. In order to obtain an accurate diagnosis, specific information on either the tumour or the patient's own genes becomes a requisite. Due to these micro-scale factors, biomarkers in certain genes can be diagnosed more accurately (Cruz & Wishart, 2006; Colozza *et al*., 2005).

There are many approaches which have been implemented on genomic datasets statistically and biologically in order to expand knowledge of the molecular basis of cancer, such as real-time polymerase chain reaction (McLendon *et al*., 2008; Fortunato *et al*., 2014). However these approaches face some difficulties such as the high dimensional nature of the data or noisy characteristics of the data (Pal *et al*., 2007). Considering the ML (machine learning) capability in high-dimensional data, it can be said that ML is particularly well-suited to implementation on genomic datasets in respect of high proteomic and genomic measurements (Cruz & Wishart, 2006).

2.2 Machine Learning

Machine learning is a scientific discipline that provides computers with the ability to distinguish patterns in datasets, even though the datasets are comprised of high-dimensional, noisy and complex data (Cruz & Wishart, 2006). Depending on the learning algorithm or implementation procedures, the machine learning area can be classified into a number of broad categories: supervised, unsupervised, semi-supervised, reinforcement learning and so on.

Artificial Neural Network (ANN) is one of the popular methods to recognise patterns in data s through sequential training procedures that will be explained in the next sections (Basu, Bhattacharyya, & Kim, 2010).

2.2.1 Artificial Neural Networks

The brain consists of a large number of inter-connected neurons that receive a number of biological signals to dendrites and pass this information on through their axon. Artifical Neural Networks (ANN) systems have been inspired by those biological systems. The main part of ANN, known as a perceptron, mimics the neurons in the brain. It receives some input values from the environment and that corresponds to dendrites part and after the received values are subjected to an activation function that that will be explained in the next section, the output value is obtained end of the perceptron that corresponds to axon (*see* Figure 2.5).

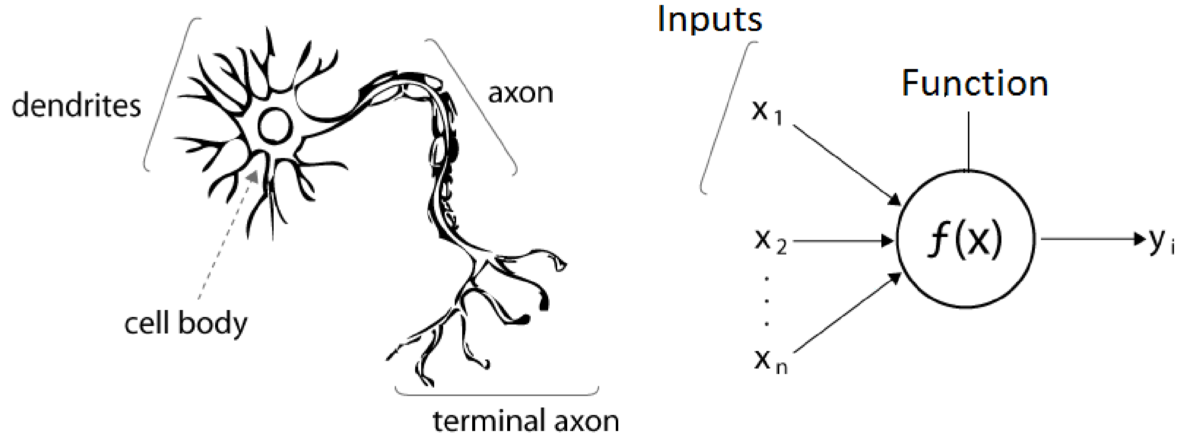


Figure 2.5: Neuron structure with dendrites, cell body and axon (left), Perceptron design, which mimics neurons, with inputs (right) (www17).

The output of neuron can be calculated as;

$$y = f(\sum_{1}^{n} x_i w_i) \quad 2.1$$

In the brain system, when information reaches the terminal axon point, it passes to the other neurons through connection areas which are called synapses. By connecting millions of neurons through the synapses, the brain is formed. An ANN in fact consists of systems have been inspired by those biological systems. In an ANN, the neurons are connected and organised in layers (*see* Figure 2.6). Depending on the number of layers, the ANN is called either a single layer or a multilayer neural network.

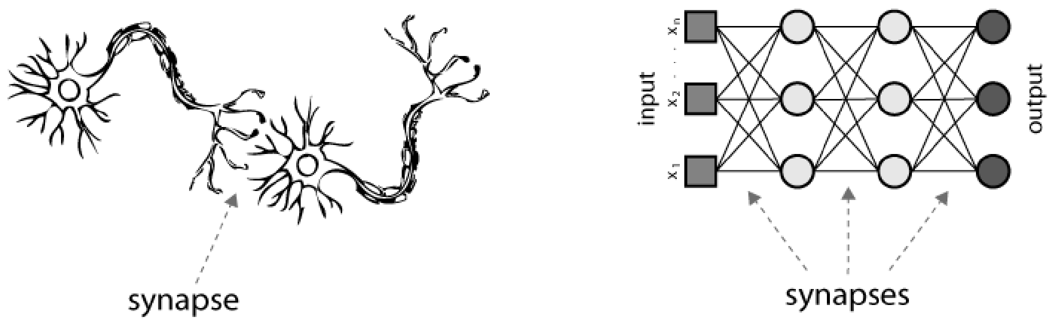


Figure 2.6: Connected neuron through the synapses for both the brain (left) and ANN systems (right) (www17)

The neural network model was first created in 1943 by Warren McCulloch and Walter Pitts (McCulloch and Pitts, 1943). The model was based on multiplication of inputs and weights (Eq 2.1) then the output is produced by a linear threshold function. However this model could only produce binary outputs that makes is simplistic model. After this creation, the next promising study was conducted by Frank Rosenblatt. Rosenblatt discovered perceptron networks by applying an extra value (bias) to the sum of inputs and weights. Rosenblatt also made weights adjustable which can be used to minimise error between perceptron and real output. Since the perceptron was able to classify a continuous-valued input into one of two classes; it made a big contribution to pattern recognition area (Hagan, Demuth and Beale, 1996).

2.2.1.1 Single Layer Neural Network

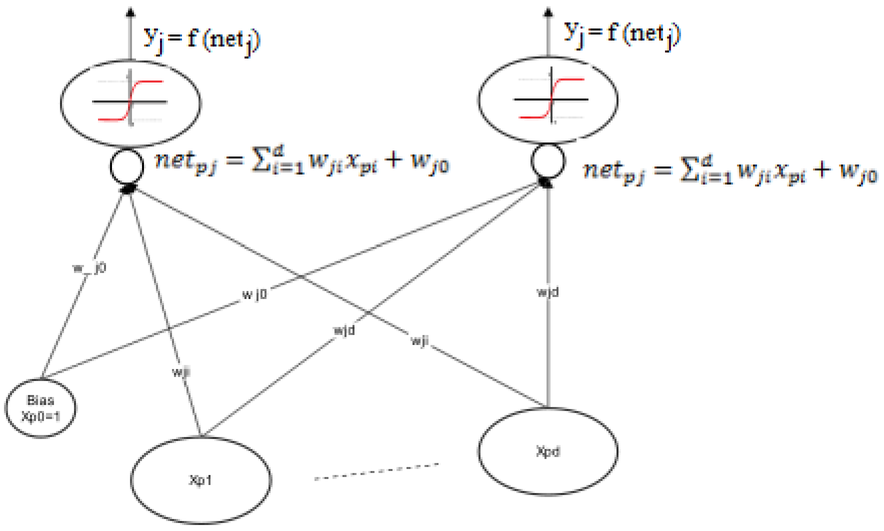
A single layer neural network basically consists of a set of perceptron and it predicts patterns by separating data linearly (*see* Figure 2.11). As can be seen in Figure 2.7, there are neurons which receive multiple inputs that are typically in the range [0, 1] and represented by x_i. The taken input values are multiplied by the value of weights (w_i) linearly. The weight is a connection value and it can either amplify or deamplify input values. In case of the linear combination net_j (Eq. 2.2) it ends up with zero, and there is another constant value which is called the bias neuron. The bias neuron can be described as a coefficient value in a function to provide consistency to the calculations. If the bias neuron value is set to 1, it does not receive any value from other neurons. During the process, the bias neuron value (1) is multiplied by the connected weight value and the result of the multiplication, which is the weight w_0, is the sum with linear combination (Eq. 2.2). After linear combination of weights and w_0, the result is included to an activation function $f(.)$ (Eq. 2.3) which is used to determine the output of neuron y_j by comparing the result and the threshold value (t). Calculations for a single layer neural network are as follows:

- Multiplication of input values and following weights;

$$net_j = \sum_{i=1}^{d} w_{ji}x_{pi} + w_{j0} \quad (2.2)$$

- The result of linear multiplication (2.2) are included to an activation function

$$y_j = f\left(net_j\right) = \{1 if net_j > t; 0 otherwise \quad (2.3)$$



$y_j = f(net_j)$
$y_j = f(net_j)$
$net_{pj} = \sum_{i=1}^{d} w_{ji}x_{pi} + w_{j0}$
$net_{pj} = \sum_{i=1}^{d} w_{ji}x_{pi} + w_{j0}$
w_j0
w j0
wjd
wji
wjd
wji
Bias
Xp0=1
Xp1
Xpd

Figure 2.7: A Single Layer Network that takes linear input vector x and multiplies it by the connected weights vector$w$to get net_j value. Finally, the output value (y_j) is decided by using activation function ($f\left(net_j\right)$)

The calculations is done by using the activation function which so far can only form AND,OR and NOT classifications. The example below shows two inputs and one output single layer;

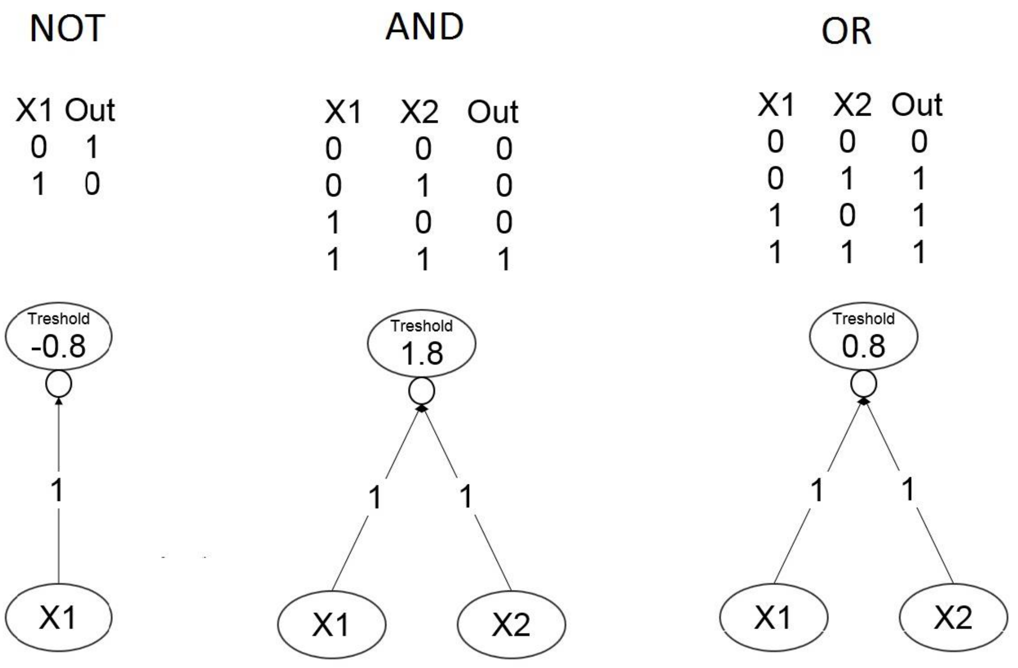


Figure 2.8: The classification with NOT, AND OR in the perceptron

As Figure 2.8 shows, an implementation such as Eqs 2.2 and 2.3 $net = w_1x_1 + w_2x_2$ and $out = f(net)$ is only capable of NOT, AND OR classification that can only separate data by drawing a straight line, as is illustrated in Figure 2.9 for AND - OR classification.

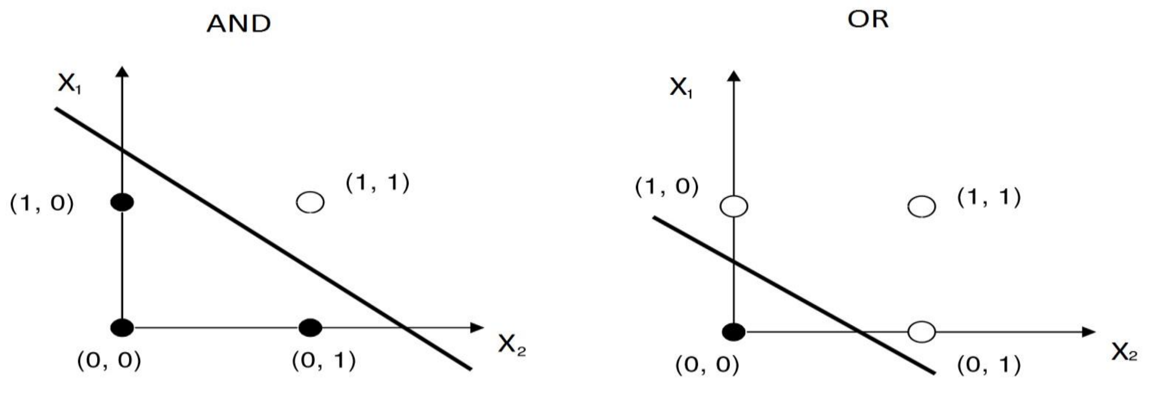


Figure 2.9: Decision boundary for AND - OR classification.

2.2.1.2 Single Layer Neural Network Optimisation

As it is shown above, the data can be classified with a straight line by the single layer neural network model. However, in the case of using further data points or high dimensional data, this line does not separate them at the first step. That is why, based on the error value between network ouput and real output, the network has to be optimised step by step. The optimisation network is implemented through the training process which is done by calculating the error of the network (Eq. 2.4) and using the gradient of descent of the error function to update the weight values (Eq. 2.9), which can eventually result in minimising the error in the network (*see* Figure 2.11).

The mathematical steps of the optimisation network are as follows;

The error function of the network is;

$$E(w) = \sum_{y=0}^{h} \left(y_j - o_j\right)^2 \quad (2.4)$$

The gradient of descent of the error function is found in order to get the direction in error minimisation (*see* Figure 2.10):

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial net_j} . \frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} (2.5)$$

This derivation function can be evaluated as;

$$\frac{-\partial E}{\partial net_j} = \frac{-\partial E}{\partial y_j} . \frac{\partial y_j}{\partial net_j} = \left(y_j - o_j\right) f'\left(net_j\right) (2.6) \quad that\ isdenoted\ by\ \delta_j$$

and derivative of $net_j withrespectweightisequalinputvalue x_i$

$$\frac{\partial net_j}{\partial w_{ji}} = x_i (2.7)$$

By combining equations 2.6 and 2.7, the gradient of descent of the error function can be written as

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = \delta_j x_i (2.8)$$

After the gradient of descent of the error is found, it is multiplied by a given value (the learning rate) to manage step length in the descending process (*see* Figure 2.10). Then the weight changing value (Eq. 2.9) is calculated by multiplying learning rate (η), δ and input value x_i ;

$$\Delta w_i = -\eta \frac{\partial E}{\partial w} = \eta \delta_j \quad x_i = \eta \left(y_j - o_j\right) f'\left(net_j\right) \quad x_i (2.9)$$

δ_j is the error between real output and the network output

Δw_i is the weight changing value of i^{th} neuron

η is a given learning rate.

Thereafter, by updating the weight values (Eq. 2.10) with the weight changing values one by one, the error minimisation is implemented (*see* Figure 2.10) and the network can converged or classified data as shown in Figure 2.11.

$w_i = w_i + \Delta w_i$(2.10)

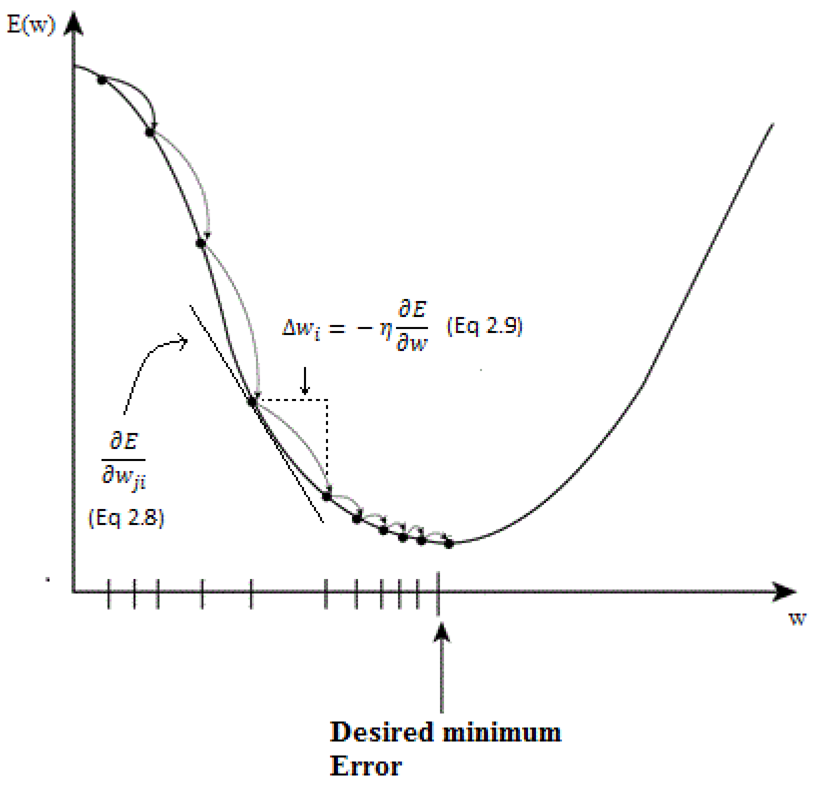

E(w)
$\Delta w_i = -\eta \frac{\partial E}{\partial w}$ (Eq 2.9)
$\frac{\partial E}{\partial w_{ji}}$
(Eq 2.8)
w
Desired minimum Error

Figure 2.10: Error descending; gradient descent of error is calculated (Eq.2.4) and each step length (delta weight) is found by multiplying gradient descent with learning rate (Eq. 2.8). The weight is updated according to the found delta weight (Eq. 2.9) and finally desired minimum error is reached by repeating those processes.

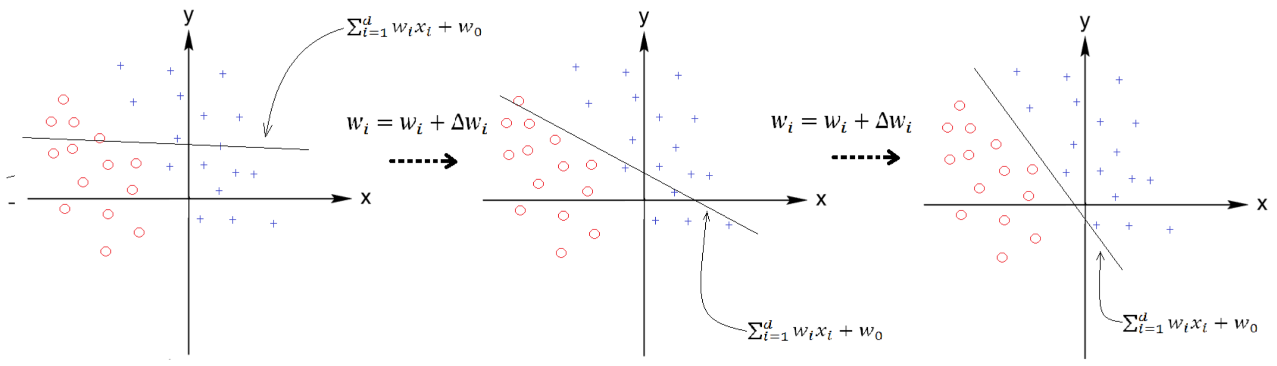

y
x
$\sum_{i=1}^{d} w_i x_i + w_0$
$w_i = w_i + \Delta w_i$
y
x
$\sum_{i=1}^{d} w_i x_i + w_0$
$w_i = w_i + \Delta w_i$
y
x
$\sum_{i=1}^{d} w_i x_i + w_0$

Figure 2.11: Single Layer Neural Network Data Classification; after each weight updating steps, the error of model (distance between data and straight line) tends to reduce. By reaching the desired error value, the data is separated as requested.

Although the discovery of single layer neural networks made a big contribution to the pattern recognition field, it had some limitations (Minsky & Papert, 1970). Minsky and Papert stated that a single layer neural network is incapable of XOR classification (Minsky & Papert, 1970). XOR is a boolean classifier which is true for two variables if and only if one of the variables is true and the other is false, which provides the ability to obtain a non-linear result in a neural network and it can only be implemented with more than one layer of neurons due to its requirements. Although the fact that it was incapable of XOR led to the discontinuation of neural network research for a time, the advent of multilayer neural networks and backpropagation provided XOR implementation and ANN research was recommenced (Rumelhart & McClelland, 1986; Hagan, Demuth & Beale, 1996). It is worth noting that, in addition to XOR implementation, a multilayer neural network can also use other non-linear/linear functions for activation such as sigmoid, sigmoid stepwise, hyperbolic tangent, Gaussian or Elliot activation functions, which can provide better performances (Isa *et al*., 2010).

2.2.1.3 Multilayer Artificial Neural Networks

In the multilayer ANN, connected neurons are organised into layers, and even though there are a couple of connectivity designs in multilayer ANN such as a recurrent neural network, the most common design is feedforward, which was used in the present study as well. In the feedforward design, each neuron in a layer is connected to all the other neurons which are in the other layers next to it. Every neuron receives values either from the environment or from other neurons. The neurons that take values from environments are known as input neurons, other nodes that take values from different nodes are called hidden layer neurons, and the rest of the nodes that have an impact on the environment are known as output neurons, as shown in Figure 2.12 (Floreano & Mattiussi, 2008 ; Haykin, 2007).

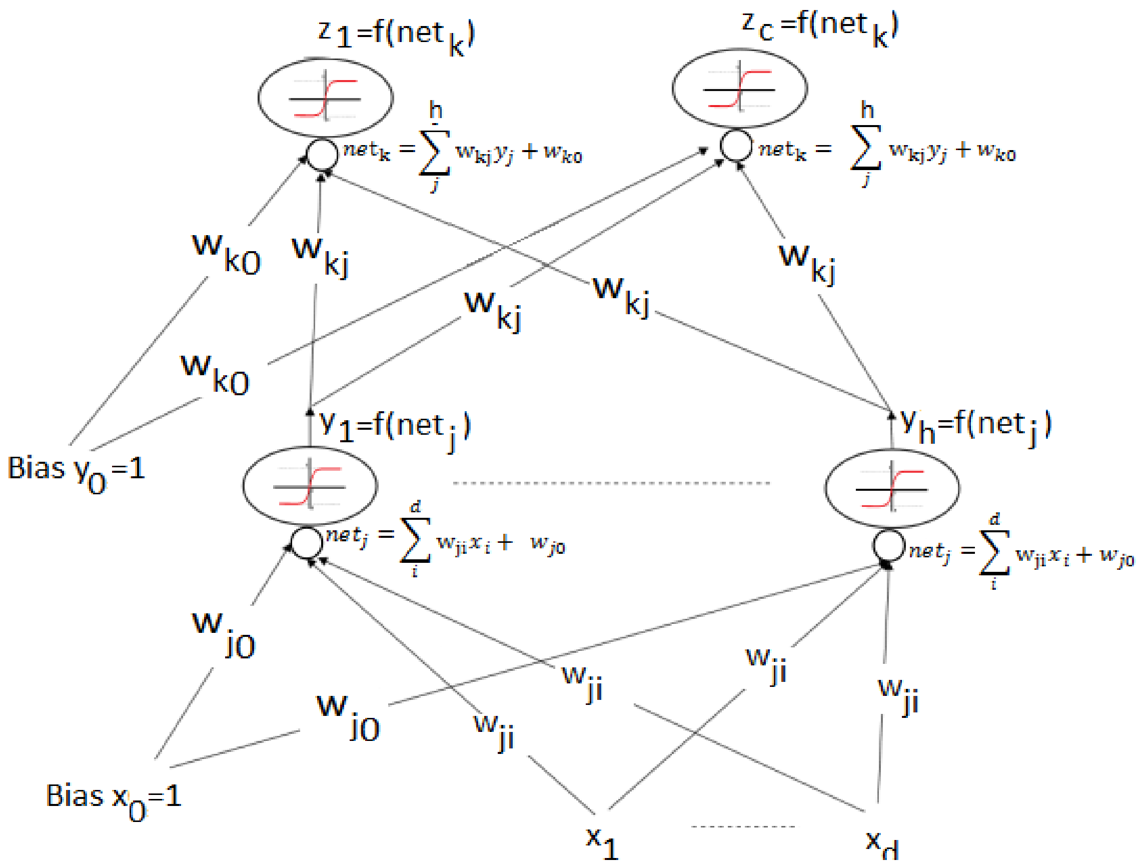


Figure 2.12: A Multilayer Neural Network Structure designed with 'd' number input, 'h' number hidden and 'c' number output neurons. Each neuron is connected to all the neurons in the layer next to it.

$x_i$is the input to the neuron

w_{ji} is the weight from i^{th}neuron (from input to hidden)

net_j is the net input to the following nodes

f_j is the activation function

$w_{j0}$is the bias value

h refers to the hidden layer neuron number

w_{kj} is the weight from j^{th} neuron (from hidden to output)

c refers to output layer neuron number

$net_k$is the net input value to the k^{th} output neuron that can be any neuron in output layer

$z_k$is the output of neuron

2.2.1.4 Feedforward propagation in Multilayer Artificial Neural Networks

As already stated mentioned before feedforward propagation is the most frequently used network structure in many applications. The propagation calculations are similar to single layer neuron network feedforward. Firstly input values are taken as a vector (x) and this is multiplied by the next layer neuron weights (w) one by one (Eq 2.11). After completing the multiplication for each hidden layer neuron (except the bias neuron), the obtained net_j values are included in the activation function f_j. Thereafter the propagation continue from hidden to output neurons using the same procedures; taking all hidden neuron values as vector (y) and multiplying them by the output neuron weights values (w) (Eq 2.13). At the end of the propagation, network output values (z) are obtained with activation function $f(.)$ (Eq 2.14) (Haykin,2007).

The mathematical feed forward propagation for Multilayer ANN is done as follow;

- Each input value is multiplied by the following weight values (w_{ji}) one by one and summed up with bias neuron values $(w_{j0}.1)$. This process is repeated for all the hidden neurons to calculate net_j values ;

$$net_j = \sum_{i}^{d} w_{ji}x_i + w_{j0} \quad (2.11)$$

- After that the net_j values are included to activation function $f_i(.)$ to calculate hidden neuron values (y_j)

$$y_j = f_i(net_j) \quad (2.12)$$

- After completing the linear multiplication and activation function (Eq 2.11 and 2.12), the hidden neuron values (y_j) are ready to be multiplied by the next following weight values (w_{kj}) and summed it with bias value (w_{k0})

$$net_k = \sum_{j}^{h} w_{kj} y_j + w_{k0} \quad (2.13)$$

- Finally, as was done before, each hidden neuron values (net_k) are put in to the activation function $f_i(.)$ again to obtained network output values (z_k)

$$z_k = f_k(net_k) \quad (2.14)$$

Once a structure is built, in order to obtain the desired output values, the ANNs are trained by adjusting the weight. In the training procedure, some values (the output of which is already known) are given to input layer nodes and the output of those inputs is compared with real output value which is represented byz_k. The comparison result gives the error $\left(\delta_k \right)$ in the network and it is calculated as:

$$\delta_k = (z_k - o_k)$$

Then, according to the calculated error, the weights are adjusted to get the minimum error as it is stated in the single layer neuron subtitle. After converging the network, the outputs of the network should be the same value as or close to the desired output values. However, optimising a multi-layer network has some disadvantages concerning the hidden layer error calculation (Russell & Norvig, 1995). Because the training data do not convey what value should be owned by hidden layers, an algorithm called backpropagation is used to calculate the error of the hidden nodes (Russell & Norvig, 1995).

2.2.1.5 Backpropagation in Multilayer ANNs

The backpropagation model is implemented backwards from output to input nodes in order to obtain the error of the nodes and to adjust weights (Bishop, 2006). The procedure of backpropagation is done by taking the derivative of the error function (Eqs 2.16 and 2.22) with respect to the connected weights and updating the weights according to the gradient descent direction.

The weight-changing steps from output to hidden layer is the same as for single layer neurons;

First, the error function is defined;

$$E(w) = \sum_{k=1}^{c} (o_k - z_k)^2 \quad (2.15)$$

And gradient descent of it is done by taking derivative with respect to weight;

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k} . \frac{\partial net_k}{\partial w_{kj}} = \frac{\partial E}{\partial z_k} . \frac{\partial z_k}{\partial net_k} . \frac{\partial net_k}{\partial w_{kj}} \quad (2.16)$$

The evaluation of this derivation can be done as;

$$-\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial z_k} . \frac{\partial z_k}{\partial net_k} = (o_k - z_k) f'(net_k) \quad which\ is\ \delta_k \ (2.17)$$

$$\frac{\partial net_k}{\partial w_{kj}} = y_j \quad (2.18)$$

The gradient is formed by combining 2.18 and 2.18 as;

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k}.\frac{\partial net_k}{\partial w_{kj}} = \delta_k y_j \quad (2.19)$$

After the derivative of weight function has been found, it is multiplied by the learning rate and finally the weight changing can be found

$$\Delta w = -\eta \frac{\partial E}{\partial w_{kj}} = \eta \delta_k y_j = \eta (t_k - z_k) f'(net_k) \qquad (2.20)$$

By following, the weight changing value is added to weight to optimise network between output and hidden layer;

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \quad (2.21)$$

So far, the mathematical procedure is the same as for a single layer neural network for updating weights. However, when it comes to weight changing between hidden and input neurons, some extra calculations are needed because the hidden layer error is a ‘black box’ in a multilayer ANN.

As was done previously, derivation of error function with respect to the belonging weight value (w_{ji}) is calculated;

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial net_j}.\frac{\partial net_j}{\partial w_{ji}} \quad (2.22)$$

However, the δ_j calculation step differs from equations 2.6 or 2.17. This time, the δ_j calculation will involve all the weights (w_{kj}) from the hidden output layers;

$$\frac{\partial E}{\partial y_j} = \frac{\partial}{\partial y_j}[\frac{1}{2}\sum_{k=1}^{c} (z_k - o_k)^2]$$

$$= -\sum_{k=1}^{c} (z_k - o_k)^2 \frac{\partial z_k}{\partial y_j}$$

$$= -\sum_{k=1}^{c} (z_k - o_k)^2 \frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j}$$

$$= -\sum_{k=1}^{c} (z_k - o_k)^2 f'(net_k) w_{kj} \qquad (2.23)$$

From the equation 2.17; $\delta_k = (z_k - o_k) f'(net_k)$ can be substituted in Eq 2.23 and the δ_j value is evaluated as;

$$\delta_j = f'(net_j) \sum_{k=1}^{c} w_{kj} \delta_k \quad (2.24)$$

Which makes weight changing is;

$$\Delta w_{ji} = \eta \delta_j y_i = \eta f'(net_j)[\sum_{k=1}^{c} w_{kj}\delta_k]\, x_i \qquad (2.25)$$

After weight changing is calculated, finally weights are updated conventionally;

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \qquad (2.26)$$

All in all, by adjusting the weight values with specific protocols, the errors can be minimised and the desired output can be obtained accurately.

Consequently, multilayer ANNs with feed and backpropagation were the answer to the limitation of neural networks and today they can handle different types of classification or pattern recognition problem (Hagan, Demuth & Beale, 1996; Kourou *et al*., 2015).

2.3 Literature Review

ANNs have been used in cancer detection/classification/diagnosis for almost the last two decades (Cruz & Wishart, 2006). This chapter investigates and partially compares previous studies that have been conducted on cancer diagnosis/prognosis in terms of data mining methods comparison, biomarker identification and survivability prediction in gene expression profiles.

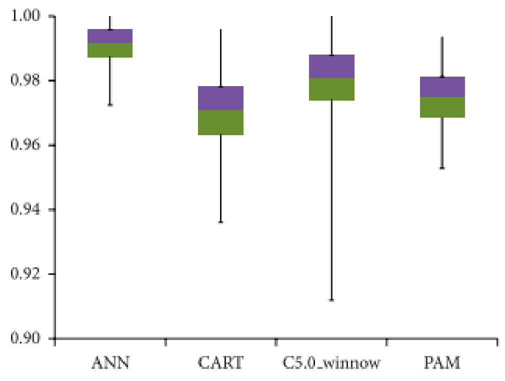
2.3.1 Multilayer ANN comparison with other data mining methods in cancer

One of the preliminary studies was that conducted by Cho and Won (2003), who attempted to precisely classify cancer by using/comparing machine learning systems on three different microarray datasets. In the first dataset (leukaemia cancer) 38 out of 72 samples, in the second dataset (colon cancer) 40 out of 62 samples, and in the third dataset (lymphoma cancer) 22 out of 47 samples were tested to classify cancer using multilayer ANNs , k-nearest neighbour, support vector machine and self-organising map. Multilayer ANNs were structured with 5~15 hidden nodes and two output nodes, and it was implemented with backpropagation. The multilayer ANNs’ recognition rates were calculated as 85.3% for leukaemia, 70.1% for the colon dataset and 69.7% for the lymphoma dataset. The results of the study showed that multilayer ANNs and KNN had the best recognition rate among other data mining methods, which shows that multilayer ANNs are quite a promising method for classifying cancer on gene expression profile compared with other data mining methods.

Another recent study was that of Chou *et al*. (2013) in which breast cancer microarray databases were used in order to predict five-year recurrence (Chou *et al*., 2013). During the experiment, three types of data mining method (ANNs, decision trees (DT) and logistic regression (LR)) were used and their findings were compared. Different datasets were pooled which had been taken from the Gene Expression Omnibus (GEO) and the US National Library of Medicine (NCBI). First, 5945 datasets were collected and filtered by Homo sapiens and breast cancer cases, without complete survival and redundancy. After choosing the same type of microarray chips (HG-U133A), four datasets were left to train. Although those databases had 922 subjects, because of missing values and effectiveness, only data from 757 patients were used in the experiment. Before merging datasets, they were pre-processed using the GC Robust Multi-array Average (GCRMA) method and R language software in order to eliminate unnecessary or irrelevant features and to combine multiple probes. During the data mining implementation process, SPSS Clementine 10.1 software was used and the ANN over-training prevention parameter was set at 80%. It was also implemented with a 20-fold cross-validation.

In order to measure the results of Chou *et al*.'s study, the term ACC (the number of correct predictions / total number of cases) was defined which presented the correct prediction of breast cancer recurrence numbers in five years. At the end of the study, the 21 most-associated genes were obtained and the ANN models displayed the best ACC score. Consequently, it can be stated that this experiment had some speciality in terms of merging datasets and comparing data mining methods. When similar studies were investigated, it was found that Sotiriou *et al*. had 34 cases and that Ivshina *et al*.'s dataset had 90 cases, whereas the present study used 757 cases (Sotiriou *et al*., 2006; Ivshina *et al*., 2006). It can therefore be said that this current experiment has a particular advantage in regard to examining a large number of cases. On the other hand, this particular advantage might be a limitation of the study depending on the classifying of the cases. Misclassified cases can cause a heterogeneity which means that irrelevant features might be grouped in the same dataset which would lead to misleading results. Despite the fact that the elimination process was carried out punctiliously, the features of cases such as phenotype definition, population ethnicity or genetic heterogeneity might not be classified correctly. Apart from dataset merging, in terms of understanding which method has a better accuracy among data mining methods, it can be concluded that this study is a good example to prove that multilayer ANNs produce better results than other methods.

Another study was conducted by Chu *et al*. by pooling microarray databases (Chu *et al*., 2014). The aim of that study was to investigate the gene expression profiles of colorectal (CRC) cancer by implementing and comparing four different machine learning methods (Multilayer ANNs, Prediction Analysis of Microarray (PAM), Classification and Regression Trees (CART), and C5.0,). At the beginning of the study, 190 different datasets were taken from the Gene Expression Omnibus (GEO). After an elimination procedure, 16 datasets comprising 1186 colorectal tumour tissues remained (53 adenoma tissues, 521 adenocarcinoma tissues, 533 primary colorectal tumour tissues and 79 hepatic tissues with metastatic colorectal tumours). Before the implementation of data mining methods, the datasets were pre-processed using the GCRMA method as had been done in the previous Chu *et al*. study. Because the 1186 cases were comprised of four different subgroups with different numbers of cases, the training data percentage was set according to the number of cases (62% for adenoma tissues, 85% for adenocarcinoma and primary colorectal tumour tissues and 52% for hepatic tissues) and an overall four-fold cross-validation was used. The data mining methods were implemented using SPSS Clementine 10.1 software. ANN was trained with a quick backpropagation algorithm on Clementine 10.1. At the end of the study, eight genes which were highly related to CRC were identified. The accuracy rates result of the study can be seen in Figure 2.13;



Test accuracy	ANN	CART	C5.0_winnow	PAM
Maximum	1.000	0.996	1.000	0.994
Q3	0.996	0.978	0.988	0.981
Median	0.992	0.971	0.981	0.975
Q1	0.988	0.963	0.974	0.969
Minimum	0.972	0.936	0.912	0.953
Mean	0.991	0.971	0.981	0.975
Standard Deviation	0.006	0.011	0.010	0.008

ANN: artificial neural network   CART: classification and regression trees
C5.0: one of decision tree   PAM: prediction analysis of microarray

Figure 2.13: Test accuracy rate in four approaches (Chu et al., 2014).

Chu *et al*. stated that multilayer ANN showed the best model stability among the other approaches. Because limited patient enrolment numbers might limit the power of machine learning approaches, this study also has similar extra features for merging databases and the number of cases , as was mentioned in Chou *et al*.'s study described above.

2.3.2 Biomarker in specific cancer type and Multilayer ANNs

Pal *et al*. (2007) conducted a study to classify childhood cancers, known as small round blue cell tumours (SRBCTs). A dataset with 2308 genes from glass-slide cDNA microarrays was used in the study. The main aim of the study was to identify biomarkers in order to enable more accurate diagnostic prediction of the four categories of SRBCT. Since a cancer type which displays cellular differences within a single cell might lead to misdiagnosis and SRBCT is one of those cancer types (heterogeneous), the possibility of misdiagnosis becomes high likely with this dataset (Heppner, 1984). The previous methods were tending to reduce genes which were irrelevant to a specific cancer type. However, those previously used methods were unable to detect subtle non-linear interaction between genes and ultimately ended up with more genes than necessary (Pal *et al*., 2007). To address this limitation, Pal *et al*. used multilayer ANNs and fuzzy clustering methods to detect the required genes accurately. The dataset in this study consisted of 88 samples, of which 63 were used for training and 25 for blind testing. Multilayer ANNs were constructed as 2308 input nodes (for each gene one neuron), four output nodes (for each category one neuron) and 150 hidden nodes and four to-detect related genes. At the first training, twenty genes were found using multilayer ANNs. In the next stage of the procedure, those twenty genes were trained and reduced to ten genes using the same method. Thereafter, the twenty first found genes were clustered using a non-Euclidean relational fuzzy c-means algorithm to compare and confirm whether the second ten genes were related to a specific cancer subgroup or not. After combining the second multilayer ANNs implementation and fuzzy clustering, the number of genes was reduced from ten to seven. At the end of study when the seven identified genes were investigated, it was seen that three of those seven genes (NAB2, LSB and EHDI) had not been identified by any other method as important (Pal *et al*., 2007). Consequently, it can be stated that the detected seven biomarker genes are enough to classify and diagnose the four categories of childhood cancers with 100% accuracy.

Although multilayer ANN could have been optimised in terms of the number of nodes, it can be concluded that Pal *et al*.'s study resulted in success in respect of its accuracy percentage and identifying three novel genes that had not been detected previously as important. In addition, double checking by using fuzzy clustering had some benefit in terms of preventing false biomarker detection.

Another study has been carried out to identify sign DNA methylation biomarkers in ovarian cancer (Wei *et al*., 2006). It is recognised that abnormal DNA methylation is a contributor to neoplasia in the human body. Correspondingly, detection of different combinations of methylated loci can be an important biomarker. Wei *et al*. (2006) focused on identifying methylated loci signs to reduce early stages ovarian cancer cases' progression-free survival (PFS). For datasets, forty advanced-stage epithelial ovarian tumours and seven normal adjacent ovarian tissues were taken from the Cooperative Human Tissue Network (Columbus, OH), the Western Infirmary and Stobhill General Hospital (Glasgow, UK), and the Cedar-Sinai Medical Centre (Los Angeles, CA). In the experiment, although there was a 811 CGI microarray at the first stage, by using Significance Analyse Microarray (SAM) for filtering, this number was reduced to a 112 CGI microarray. Then multilayer neural network and support vector machine methods, as assessed by ten-fold cross-validation, were implemented on these 112 microarray samples to identify methylated loci prognostics. As a consequence, both the multilayer neural network and the support vector machine methods showed 100% classification accuracy for the detection of different combinations of methylated loci. According to the fact that the detected biomarker has a crucial role in the accurate diagnosis of the cancer subgroup, the study's accuracy result proves that a multilayer neural network is highly effective at providing useful information to clinicians.

A further similar study that aimed to predict the occurrence of lymph node metastasis in oesophageal cancer by analysis using multilayer ANNs was that of Kan *et al*. (2004). For the microarray datasets, twenty-eight independent primary tumours samples which had been under surgery at Kyoto University Hospital were chosen. Candidate genes which had been taken from patients were included in the SAM filtering process that can statistically extract differences. By the SAM filtering, the number of genes was reduced from 8064 to 30-120. After that, multilayer ANNs were applied with feed/backpropagation and leave-one-out cross-validation. The multilayer ANNs were constructed with two hidden layers, which differed from other studies. The number of input nodes equalled the number of genes used; the first hidden layer consisted of four nodes and the second hidden had ten nodes (bottle-neck type), and the number of output nodes was set which showed the presence of lymph as 1, and 0 otherwise. After training the neural network, the results showed that the predictive accuracy of lymph node metastasis was 77% over 120 filtered genes. When the neural network was tested with 60 genes, the accuracy rate extended to 86%. In the light of these results, it can be concluded that multilayer ANNs can predict the occurrence of lymph node metastasis in oesophageal cancer to a certain extent. Since the study had two hidden layers, it can be an indicator for the prediction ability of networks with extra hidden layers. Also, this study shows the advantages of SAM filtering for the gene filtering processes.

Another study by Abd El-Rehim *et al*. (2005) aimed to analyse protein expression in categories of breast cancer tissue samples by applying immunohistochemistry (IHC) and the ANN classification method (Abd El-Rehim *et al*., 2005). A sample of 1076 breast cancer cases from the Nottingham Tenovus Primary Breast Carcinoma Series were tested in order to detect biomarkers which are indicators of breast cancer. At the beginning of the study, IHC was applied to tissue microarrays (TMA) to cluster samples formulated in terms of similarity. Further analyses of the cluster data were carried out using multilayer ANNs. The ANN architecture was built from 13 input, 60 hidden and 6 output nodes. The multilayer ANN categorized the cases into groups and examined the driving biomarker in each group. Using these methodologies, six main clusters were identified by IHC, as shown in Figure 2.14, and the multilayer ANNs could predict 1, 2, 4, 5 and 6 classes with 100% accuracy whereas class 3 was predicted with 99.57% accuracy. When all the classes were analysed by the means and the standard deviation (SD) of the mean of expression of the markers in each group and multilayer ANN, it was found that a discriminating marker which is related to epithelial cell lineage, differentiation, hormone/growth factor receptors and gene products could be detected successfully. As a result of this study, Abd El-Rehim *et al*. (2005) stated that two of the important discriminator proteins (CK18 and CK5/6) identified were related to the mammary gland anatomy and cellular structure of its parenchymal tissue (Abd El-Rehim *et al*., 2005). Consequently it can be said that the multilayer ANN method with IHC clustering can give essential clues to tailoring the treatment of breast cancer.

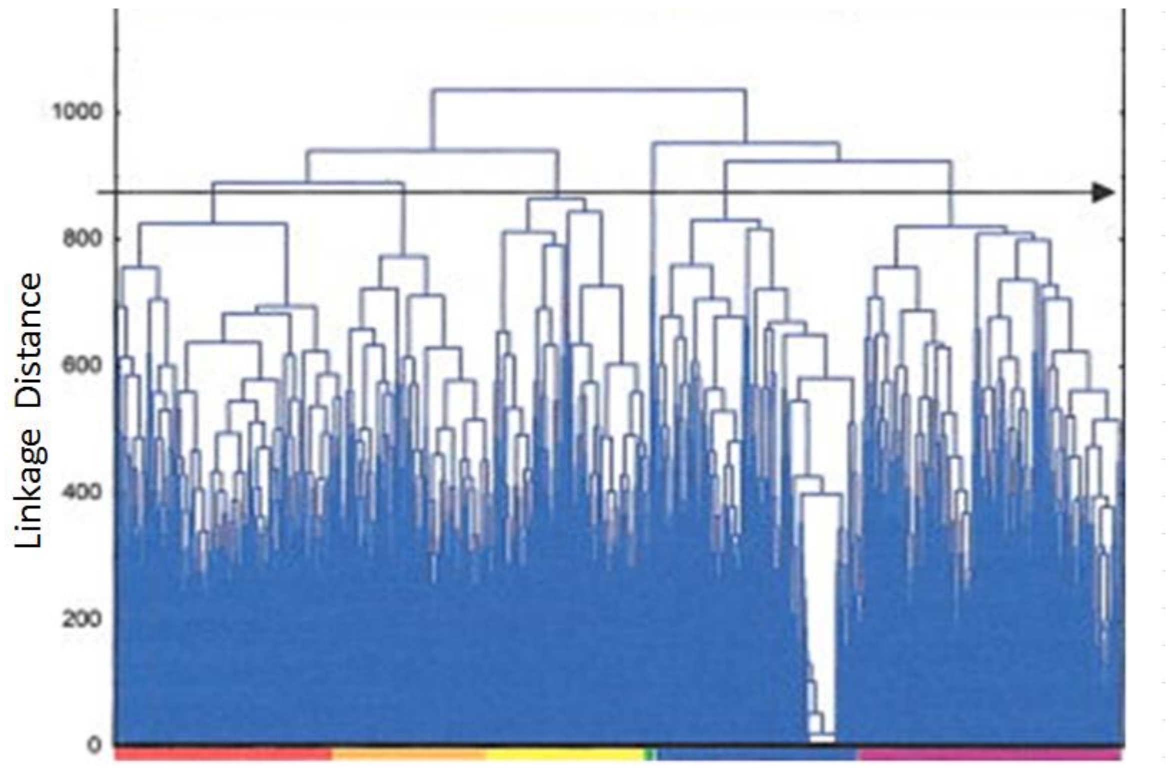


Figure 2.14 Cluster tree diagram of a tumour sample. Clusters are arranged from left to right, starting from cluster 1 and ending at cluster 6.

A subsequent study was conducted by Agarwal *et al*. (2014). Since there had been great interest in Ki-67 as a proliferation marker with both prognostic and predictive value in breast cancer, the main aim of their study was the identification of common genes predictive of Ki-67 expression in three different microarray breast cancer datasets by using multilayer ANNs with feedforward, backpropagation and cross-validation. It should be noted that unlike most other studies, another cross-validation method, which is named 'Monte-Carlo' was used in this experiment. The Monte-Carlo cross-validation method has the ability to avoid over-fitting of the data and is more steady than leave-one-out cross-validation (Xu, Liang & Du, 2004). The multilayer ANN consisted of two hidden nodes and the sigmoid function. The datasets were collected from the Nottingham breast cancer microarray (training set), the Uppsala breast cancer cohort (test set) and the METABRIC cohort (validation set). By using ANN, the top 200 probes for Ki-67 status were identified for each dataset. As a second phase, a non-reductionist approach network growth strategy was implemented in order to make the Ki-67 system more precise in breast cancer analysis. In this approach, the open-access online database Search Tool for the Retrieval of Interacting Genes/Proteins (STRING, Version 9.1) was used to determine linkages between the top ten proteins interacting with Ki-67 and in turn between each of the top ten and their top ten genes, as shown in Figure 2.15. Thereafter, those findings were compared with the results of the first phase in order to identify the common genes. As shown in Figure 2.16, 64 unique genes were identified across all three cohorts.

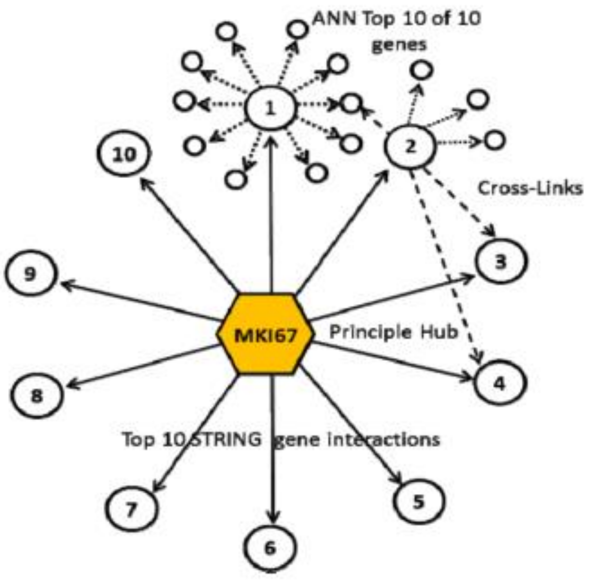


Figure 2.15: The linkages of top 10 for Ki-67 (Agarwal *et al*., 2014) breast

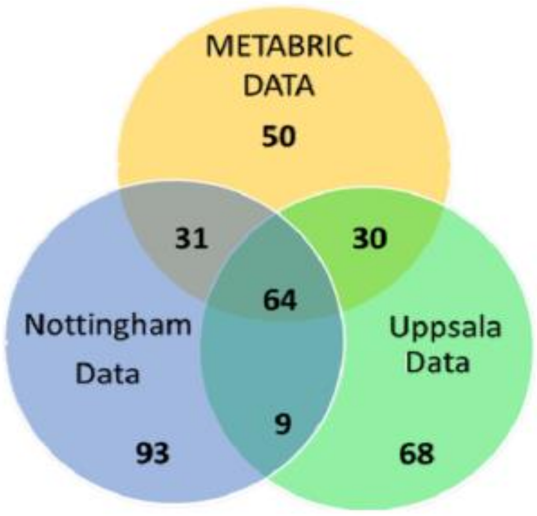


Figure 2.16: Depicted Ki67 common genes in 3 cancer dataset (Agarwal *et al*., 2014)

Agarwal *et al*. (2014) stated that in a comparison of the 64 genes with previous studies, there was a significant overlap with other similar signatures such as CEP55, CENPA, CENPE and spindle checkpoint proteins such as TPX2, AURKB, CDC20, which might contribute to tumour progression. However eight of the twenty most common genes for KI-67 were not listed among these 64 genes (Agarwal *et al*., 2014) so it can be concluded that the study result could have been more precise. Additionally, the ANN approach could have been used rather than STRING to predict associations for Ki-67 due to the flexibility of ANN.

2.3.3 Predict survivability in cancer by using Multilayer ANNs

A study by Chen, Yang and Chiu (2009) sought to predict survival time in diffuse large B-Cell lymphoma (DLBCL), follicular lymphoma (FL) and ovarian cancer patients by using multilayer ANNs (Chen, Yang & Chiu, 2009). The datasets were obtained from Shipp's study for DLBCL, as 58 untreated patients, Dave's study for FL, as 95 untreated patients, and from Duke University Medical Centre and the H. Lee Moffitt Cancer Centre and Research Institute including 69 deaths from ovarian cancer (Shipp *et al*. 2002; Dave *et al*., 2004). The gathered dataset was pre-processed using BRB-Array Tools. Then, in order to optimise the ANN architecture, the dataset was divided into two groups of 90% and 10%. The 90% of the data group was used for training with a commercial software (STATISTICA version 8.0) and the rest of the data (the 10% group) was used for testing. During the training process, inputs were initialised randomly and at the end of the calculation the obtained output was compared with known survival time. The process was then repeated by altering the weights between nodes until the error was reduced to a negligible rate. To optimise ANN, it was tested with 5 to 30 hidden nodes. However, the multilayer ANNs can overfit the training data rather than decrease the generalisation accuracy, so the multilayer ANN's architecture was determined by trial and error. At the end of study, the result was calculated as follows: for the DLCBL data set, differences between values and estimator **(**RMSE) were identified as 2.68, and the linear relationship between values and estimator (correlation coefficient) was calculated as 0.956. Additionally, some genes (D63879_at (KIAA0156), HG1879-HT1919_at (ARHQ), U41815_at (NUP98) and X77366_at (TCF11)) which are related to cancer were reported. For the FL dataset; RMSE was calculated as 27.69, and correlation coefficient was 0.771 whereas these values were 17.23 and 0.868 respectively for the ovarian cancer dataset as shown in Figure 2.17 for the three datasets.

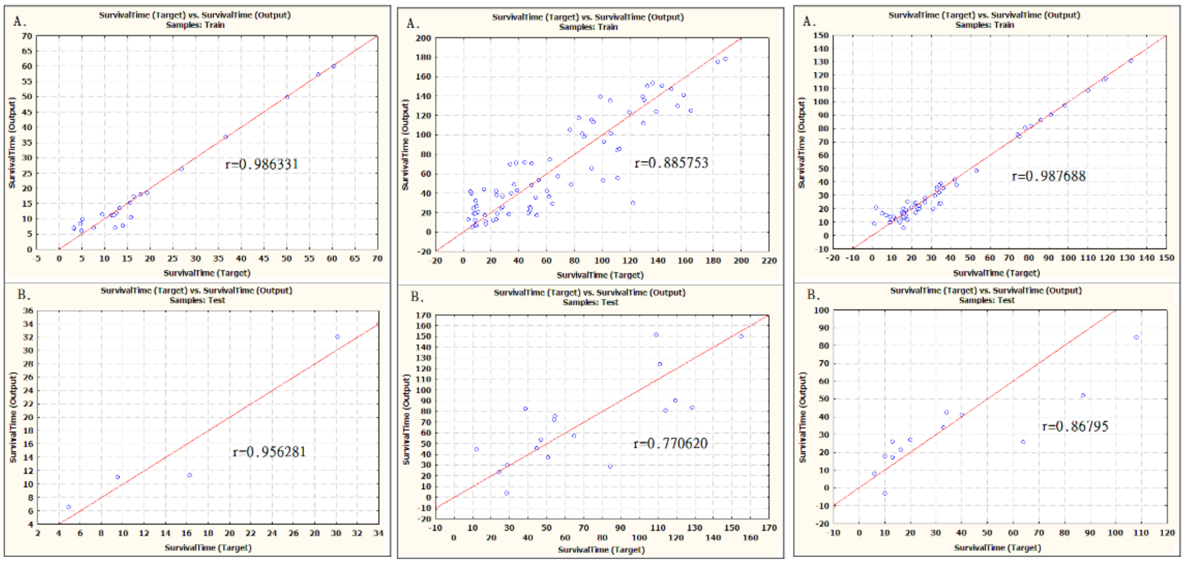


Figure 2.17 : DLBCL (left) survival time observation and prediction results , FL (middle) survival time observation and predictions tresult, Ovarian cancer (right) survival time observation and prediction results (Cheb, Yang andChiu., 2009)

Chen, Yang and Chiu (2009) therefore declared the prediction of ANNs to be accurate and acceptable. However, the findings could have been improved by comparison with other study results as was done by Agarwal *et al*. (2014).

Another similar study was conducted with validating gene expression profiles in order to obtain clinical outcomes of breast cancer (Lancashire *et al.*, 2009). The study hypothesized that a gene expression signature would be capable of predicting survival with some accuracy. Considering the difficulty of breast cancer heterogeneity, multilayer ANNs with backpropagation and Monte-Carlo cross-validation methods were implemented. In the multilayer ANNs' structure, the number of hidden layer nodes was restricted to five and the output nodes gave evidence of metastasis based on a YES/NO sigmoidal function.

Research has been done on Van't Veer's dataset which used breast cancer disease samples. In the experiment, 78 samples were used initially and were divided into 60% for training, 20% for validation and 20% for blind testing independently. Consequently, nine genes were obtained that prognosed with 98% sensitivity. Since the capability of predicting metastases has a high degree of accuracy, in a further stage, another 295 patient cohort dataset (NKI295) was validated. The obtained nine gene signatures were applied to classify this series of cases. The discovery was made that the data were able to be divided into the proper groups which had been defined by the original 70-gene signature. Consequently it can be observed that this study resulted in a high degree of accuracy and affirmation of these findings.

This chapter has investigated previous relevant studies in terms of the comparison of data mining methods, biomarker identification and survivability prediction in gene expression profiles using a multilayer neural network. It can be concluded that a multilayer neural network is the most plausible method among other data mining methods for the classification/prediction/identification of cancer cases accurately, which proves that this method would be helpful for uncovering hidden patterns in a dataset that can help clinicians in their decision-making.

Chapter 3

Requirement and Analysis

This chapter will first describe the requirements of the project and then the means of overcoming the challenges will be explained in the analysis part. The test results of the tool and the evaluation process will also be explained.

3.1 Requirements

As mentioned in section 1.1, the main aim of this project is to develop a computational tool that will be trained by cancer datasets, and after training it will be able to predict information such as metastasis capability, and the stage and/or typology of cancer depending on the training dataset.

The requirements of this project can be split into three categories; functional, software implementation and non-functional.

- First, the dataset that will be used in the project should have the requisite qualifications such as completed input and outputs values or genes that are highly related to cancer subtypes. In order to classify cancer accurately the tool should be trained with genes that are related to specific cancer types. Considering that there are plenty of publicly available gene expression datasets on the web and that these datasets include variety of genes/features, we should ensure that our dataset consists of the necessary genes. Therefore a small blue cell tumour (SRBCTs) dataset which was part of Khan *et al*.,(2001) study is used in the present project.
- Second, as can be seen in the literature review, generally studies have used SPSS or other commercial software in multilayer layer neural network training. Considering that our project network will be licensed under BSD3 to provide the ability of re-structuring in terms of calculation methods or layer/neuron numbers, commercial/non-commercial products are not used for the implementation of a neural network.
- Third, the tool should have high prediction accuracy in cancer. However, machine learning systems are prone to overtraining (the tendency of fitting data excessively), which causes a decrease in prediction correctness. Therefore a technique called cross-validation has to be performed in the training process to avoid overtraining and to obtain high-accuracy results.
- Due to the need for the tool to be robust, unit testing should be implemented on codes to ensure that process is carried out properly, and also the code should be tested by a performance analysis tool to detect and fix any code bottlenecks.
- Additionally, after completion the testing of the tool, it will be released as a library called Cunernet with a BSD3 licence to reach the widest community. In that way, the tool might be made use of by any communities or contributors who might wish to be involved in the further development process.
- Furthermore, the latest standards of C++ language should be performed because the tool will be implemented on CUDA using the Thrust library.
- Finally, since the tool will be used by other users, compiling and verifying code changing should be done before uploading any written code.

The functional requirements and software implementation and the non-functional requirements can be outlined as shown in Table 3.1

	Functional Requirements
1	The tool should predict cancer information(metastasis capability, stage, topology) with high accuracy (95% or above preferably)
2	The dataset should only have the required genes/features
3	The used libraries should be under BSD-like licence.
4	The tool should be flexible in terms of implementing different learning rules and updating protocols, neuron numbers in layers or input/ output vector size.
5	By implementing unit testing , the code should be made robust
	Software Implementation
1	The tool should be developed by using best practice developing techniques to release as an open source
2	The user should be capable of deciding the implementation (training/testing) and the network structure (multilayer/single layer)
3	The basic network values such as learning rate, momentum or input vector size in data set should be passed by the user
4	The developed tool will be released named as the Cunernet library under the BSD licence.
5	Cmake cross-platform compilation software should be used to manage the building process of the tool.
6	The Thrust library should be used in regard to implementation codes on CUDA as a project extension
6	The Cunernet has to be implemented as self-contained shared library to be included in other projects
	Non-functional Requirements
1	The code should be analysed in terms of performance by using performance analysis tool (gperftools) and according to analysis result it should be optimised.
2	The Operating System of machine should be Linux 64-bit with installed CUDA 7.0
3	The tool should be compiled using g++ 4.8

Table 3.1 The Requirements of The Project

3.2 Analysis

By analysing of the requirements of the project, the following stages should be followed in the project:

3.2.1 Dataset

The used SRBCTs dataset, which is publicly available, includes four distinct categories (rhabdomyosarcoma (RMS), Burkitt lymphomas (BL) subset of NHL, neuroblastoma (NB), and the Ewing family of tumours (EWS)) and consist of 88 samples (25 RMS, 11 BL, 18 NB and 29 EWS). Each sample includes 96 genes, which are highly related to those four categories mentioned above.

Khan *et al*.,(2001) split the88 samples into training (63) and test sets (25). After training their network, the 25 test samples were classified with 100% and diagnosed 23 over 25 sample could be diagnosed correctly (2 samples Test Sample 10 and Test Sample 20 were not diagnosed).

Due to the 96 genes high relevance to the SRBCT cancer types, being publicly available and showing promising result, the SRBCTs dataset will be chosen to train and test our network.

3.2.2 Programming Language Choose

Due to the flexibility and licence issues, own Multilayer Neural Network software will be developed in C++11 language. Concerning that the written code will be optimised by Google Performance Analysis tool to achieve a better performance, C++ was a proper choice. Additionally as an extension of project the software will be implemented on GPU by using Thrust library in the future, it can be said that C++11 was the most proper language to develop our tool. In the light of C++11 and Google Performance tool using requirements, the project environment is chosen as Linux 64 bit operating system with CUDA7.0

3.2.3 The Network Design

In the present study the structure of the network will be limited to linear perceptron, but the software will be designed to be extended to multilayer neural network depending on passed information from the user. Also the learning algorithm of software will be implemented same as Khan *et al*.,(2001) (updating weights every 10 samples) and in addition that, different optimisation algorithms such as batch and stochastic learning will be added to software code as an optional choice.

3.2.4 Training Network

In the training procedure of the network, 63 samples over 88 will be taken to train our network and three-fold cross validation will be implemented in order to avoid over training, as was done by Khan *et al*., (2001).

The procedures for training with cross validation will be;

- First the 63 sample dataset will be shuffled and split into three equal part
- After that, any two of three parts will be taken as a training set and the remaining 21 samples are used as the validation set.
- First, the training set (42 samples) will be subjected to training procedures and the other 21 validation set, will be kept out of training.
- The training procedures will be repeated 100 times with the same training and validation datasets.
- After 100 times completion, another 21 samples from the dataset will be chosen as a validation and the other 42 samples will be used as training.
- The 100 times training procedures will be repeated again with the new chosen compound of 42 samples.
- These steps will be repeated three times and each time a different fold (21 samples) will be chosen as validation set. By doing that over-training in network will be prevented
- End of these stages one iteration will be completed and by shuffling the 63 samples again iteration will be started.

3.2.5 Best Practice Development

Owing to the intention to release the tool an open source library, the best practices of open source development techniques will be used. All code development processes will be done on the web application hosting service Bitbucket. To manage code changing on Bitbucket, the “git” revision systems will be used and each code changing steps will be uploaded to the Bitbucket page with an explanatory note to be a guide for the history of the code development or a guide for further studies. Also, in the case of getting an major errors, the project codes will be able to be rolled back by tracking the history on Bitbucket.

Due to the fact that documentation is an important way to communicate with other developers, the code comments will be written base on Doxygen rules and The Doxygen tool will be used to generate documents as html or pdf automatically which will be found in the *doc* folder on the Bitbucket project page. The generated document will be an easy guide to examine code in regard to functions or variables for potential contributors and also for the current author in the future.

In order to deal with compiling source code fields in different directories, the CMake cross-platform software will be used to manage the building processes. Additionally, by providing readme files will be written to guide how to compile/run The Cunernet that files can be found in the project folder on Bitbucket.

3.3 Testing and Evaluation

In this section, the present study classification and diagnosis method will be described rather than Khan *et al.*, (2001) classification/diagnosis method. However in section 5.4.1 Classification and 5.4.2 Diagnosis Khan *et a.*, (2001) methods will be described and implemented along with the present study classification/diagnosis method.

3.3.1 Testing the Correctness of the Training Set

As described above, 63 samples will be used to train the network and after each training dataset was trained, its correctness will be calculated. The correctness calculation will be done by comparing real output data and the network output data for each sample.

During the correctness checking, training set samples will be subjected feedforward propagation and then the network output vector will be obtained one by one. (In our case the network output vector has four elements with regard to the four cancer categories is searched) After obtaining output vector, the highest value of vector will be checked whether it has 98% weight in sum of all four values. In the case of the percentage being 98% or more, the index of this value will be compared to real output highest value index. If both indexes are the same, that means the network classified the sample correctly and correct number will be iterated with one. These processes will be implemented to all samples in the training set and after finishing all samples, the correctness percentages of training set will be calculated by dividing correct match and the dataset size number (Eq 3.1). The pseudo code of these processes can be seen in *Pseudo Algorithm 1 Testing correctness of a data set.*

$$Correctness\ percentage\ =\ \frac{correctNumber}{Dataset\ Sample\ Number} \quad (3.1)$$

3.3.2 Avoiding Over-Training

Furthermore, as it mentioned in the Analysis part, three fold cross validation will be used to prevent over-training network. Avoiding over-training will be done as follow;

- A network model will be defined for testing
- At the beginning of the iteration, the dataset (63 samples) will be shuffled and split into 3 fold
- As it is explained in 3.2, the two folds (training set) will be subjected to training process and the other third fold will be kept as validation set (without training network)
- End of this process the MSE (i.e. mean square error) for both training set and validation set will be calculated
- This training and MSE calculations will be repeated 100 times and the MSE values obtained will be stored to the vectors.
- At the end of 100 times, two vectors within the 100 MSE value elements will be obtained and written to a file
- Then the network model is set as a new beginning and the dataset will be split again after shuffling uniquely.

By repeating these steps 50 times, a 100 vectors (50 for training,50 for validation) that each of them has 100 MSE values will be obtained and plotted (see Figure 5.6) by using python script which can be found in the script folder on Bitbucket project page. By plotting a graph, the existence of over–training will be tested and, in the case of that the validation errors have a decreasing trend, it will be ensured that there is no sign for over-training in the network.

3.3.3 Unit Testing

As a part of unit testing, a relatively small dataset (4 inputs and 2 outputs) will be created manually. This dataset will be given to the network for training, as shown in Figure 5.2. During the training processes the used input vectors, network outputs values, network weight changings will be written files and checked manually to see whether process/multiplications are done as required.

When the network outputs and real outputs comparison file will be examined, it should be seen that the network outputs values are close to the real output values in each iteration, which means that the network would be converging as required.

3.4 Testing Network Correctness/ Network Evaluation

During the training process, the used network models will be saved to files which can be found in data/models directory on Bitbucket page. After completing the training of the network, the proper network model will be reloaded from one of those model files to test network correctness with the test dataset. In the current study, the test dataset consists of 25 samples with 5 “noise sample” (that is not belonging to one of the four cancer categories).

In network testing, each sample will be given to the reloaded network and subjected to feedforward propagation one by one. After this feedforward propagation, the network output vector will be obtained and its classification/diagnosis will be done according to a percentage weight in the sum of all elements, as explained in Section 3.3.1. (The percentage weight number will be decided after trials) For each correct diagnosis, the number of correct samples will be iterated with one and at the end of all samples the correct number will be divided by the test dataset sample number (25) to gather the percentage accuracy of the network. After obtaining the network accuracy, using a list of correct classified/prognosis samples, it will be compared with the results of Khan *et al*. (2001) to understand whether the present study network works as requested, and whether it is better or worse than other study networks.

The pseudo code of testing correctness can be seen below;

Pseudo Algorithm 1 Testing correctness of a data set

```
1:procedure:checkOutputCorrectness
2:
3:    for each inputVec , realOutputVec in DataSet do
4:
5:        feedforward(inputVec);
6:
7:        outputVec <- getOutput();
8:
9:        sum <- sum(outputVec);
10:
11:       highest <- getHighestValue(outputVec);
```

```
12:
13:         **if** highest > sum *0.98 **then**
14:
15:             **if** index in outputVec == index in realOutputVec **then**  correctNumber++;
16:
17:     correctness = correctNumber/ DataSet.size;
```

After the network correctness has been approved by comparison with the results of Khan *et al*. (2001), the other study cancer datasets will be used to affirm the correctness of the network training and testing.

3.5 Profiling/Performance Evaluation

The bottleneck of written serial code will be tested/monitored by using the performance testing tool and Linux *time* function. The Google Performance Analysis CPU profiler, that can be found on https://code.google.com/p/gperftools/, tool will be used to monitor unit time for each function/method relative to total unit (calles) of software. According to the gathered results from gperftools, the code parts will be changed/replaced to optimise it and after each changing the function *time* will be used to check total wall time reduction. Due to the total wall time may change depends on hardware current work load, the software will be run several times and average of wall time will be taken to compare reduction time.

In addition, the total spent time against neuron numbers will be tested on both single and multilayer neural networks and the effect of neuron/network structure on the spent time will be examined by plotting graphs.

Also it should be mentioned the intention that the Thrust library will be used to implement the software on CUDA in the future, regarding to the compability of Thrust library, the gperftool is chosen for performance analysis.

3.6 The Library Cunernet

The Cunernet is a library that can be downloaded from the Bitbucket project page and by typing simple commands it can be compiled/installed;

In order to compile and install;

- ./cmake.local

In order to compile and run;

- source local/share/CUNNET/CUNNET.conf
- ./cmake.test

The library has to be compiled using g++ 4.8 under linux 64bit to ensure compatibility with Cuda 7.0. To contribute it, the functionalities have to be added to the /src/cunnet directory and the cmake file (/src/cunnet/CmakeLists.txt) has to be edited by adding a new source file name. It is good practice to create a new test for each functionality implemented. Every new test has to be coded as the template used in "/src/tests/test_main" and ./cmake.test script file should be modified.

Chapter 4

System Design

This chapter describes chosen design techniques along with both the benefits and drawbacks.

4.1 System Overview

The project design parts can be divided in different main steps as pre-processing/choosing dataset, developing design in terms of using feedforward, backpropagation and updating weights protocols in serial code and also software design.

4.2 Selecting and Pre-process of Dataset

A gene expression dataset is a repository consists of microarray gene expressions that is explained in *2.1.2*. Today there are many gene expression cancer datasets publicly available on the web thanks to projects. “The Cancer Genome Atlas” (www16, 2015) and “Gene Expression Omnibus” (www17, 2015) can be listed as two of the main projects for creating datasets. Basically a specific cancer dataset is created by collecting variety of genes/features, which have affinity with specific cancer types, from cancer patients’ microarray sets and collected data are stored to a database properly. After the creation of a cancer dataset, it might published online and can be accessed on the project web pages.

Even though a dataset’s genes are related to cancer categories, the high number of genes (being high-dimensional) can pose some challenge for the accuracy of cancer category detection. For instance, whilst those extra features might mislead the machine learning calibration, they can also cause an overwhelmingly inaccurate calculation in the system as is shown in Figure 5.7. Consequently, the training process might be concluded as having a wrongly calibrated network which is not capable of detecting cancer subtypes accurately (Yu & Liu, 2003). Therefore, depending on the characteristics of the dataset, it should be pre-processed; however there are different methods for pre-processing a dataset to find the exact features and each of them might result in a relatively different list of genes. Bearing in mind the possibility of differences on the gene list, using the identical SRBCTs of a childhood dataset from Khan *et al.*’s (2001) study would be the correct choice to test the present study’s results and improvements. As discussed briefly in Section 3.2, there were 88 samples with 6567 genes at the beginning of pre-processing in Khan *et al.*’s study. As a first step of pre-processing, these 6567 genes were filtered using image analysis on gene expression profiling, as was shown in Figure 2.2. After implementing the red/relatively red intensity filtering method, which shows the relative rank of genes and cancer subtypes, 2308 genes remained and then a principal component analysis algorithm (PCA), which reduces the dimensionality of data, was applied to the remaining 2308 genes. Finally ten components were left to train/calibrate their neural network, and as result of that study (Khan *et al.*, 2001), 96 genes which were relevant to four cancer subtypes were found.

The stages of pre-processing can be seen in Figure 4.1

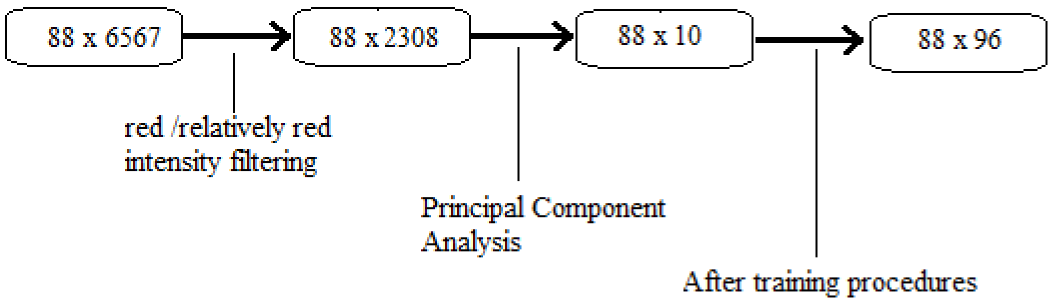

88 x 6567
88 x 2308
88 x 10
88 x 96
red /relatively red intensity filtering
Principal Component Analysis
After training procedures

Figure 4.1 Pre-processing dataset; 6567 genes are filtered by image analysis filtering ,the remained 2308 genes data dimensionality is reduced by principal component analysis and then 96 genes were found by (Khan et al., 2001) 96 genes have been found.

In the present study, the extracted 96 genes will be used as dataset in order to train and test our neural network. By doing that the results of project can be compared and possible improvements will be identified.

4.3 Developing Design

First, the programming languages will be considered for neural network development. Even though pyhton language has some libraries such as PyBrain for neural network, C++ will be more suitable choice for development on both serial and parallel algorithms due to the project will be implemented on CUDA by using Thrust library and also for the reason of using Google Performance tools to optimise code.

The development tool that will be named the “Cunernet”, will be released as an open-source library. Therefore software development best practice techniques will be used. The development process will be practised on the Bitbucket web application service by using “git” which provides opportunity to roll back the code by tracking its history. Additionally, the code history and tool itself might be used as a guide for further developments with regard to the tool will be released as open-source on Bitbucket page. Also, the codes’ comments are written base on Doxygen rules, and it is used to generate documents of code as an html or pdf file by using the *make-doc.sh* file which can be found in the data folder on Bitbucket project.

In reference to the extensibility of the software design, even though the network will be run as a single layer neural network in the present work, it will be capable of extending to a multilayer neural network depending on the typed information from users. In addition, as backpropagation weight updating protocols, weights will be updated every ten samples (as Khan *et al*. (2001) did), but two of the main learning algorithms (stochastic and batch) will be also added to the software code as an optional features. The pseudo-code backpropagation protocols can be seen in *Pseudo Algorithm 5 Updating Weights;*

Regarding the compilation of the Cunernet library, the Cmake open-source software will be used, which can compile several programming files from a tree-type directory. The compilation procedures will be written in a readme file and will be found on the Bitbucket project page.

After finishing the developing, a unit test will be performed to determine whether the software is working as requested. The unit testing implementation will be done by creating a small dataset, with four input and two output element vectors, and the output of the network will be checked during the training processes with the created dataset.

4.4 Software Design

The software should be run on a 64-bit Linux operation system with CUDA 7.0. After downloading the Cunernet software from Bitbucket, the user will be able to compile and run the software with basic commands that are written in the readme file on the Bitbucket page.

The files will be designed as a tree directory. A folder *src* will contain the source code files, a *data* folder will include datasets or written files that are created during training/testing. Also. *local* will be the target installation folder in which Cmake outputs binaries will be stored. In addition, scripts for drawing line graphs and so on will be located in a *script* folder. All the folders can be found on the Bitbucket project page.

When the software is compiled and run, a command panel will ask for some information such as training/testing, network structure (single/multilayer), dataset file name to train the network, gene number in dataset and learning rate with momentum values

In this step, the Cunernet software will be able to be trained/tested by any suitable dataset by passing different dataset file names and gene numbers into the given file.

4.5 Neural Network Design

As mentioned in the Chapter 2, neural networks are a suitable machine learning systems for detecting cancer subtypes accurately. The main restriction of this system can be stated as that

- it requires hardware which should be able to do high computations
- Regarding to high computations requisite, obtaining results may take relatively long time

On the other hand by optimising algorithms and by reducing dimensionality of dataset with correct pre-processing methods, the time can be reduced significantly and an accurate result can be gathered in favourable time.

4.5.1 Feedforward Algorithm

During the network building process, weights will be initialised randomly from a certain range. The range of weight values can be expressed as [r,-r] and it will be able to be calculated as follow;

$$r = \frac{0.1}{F_i} \quad 4.1$$

Fi: Number of connected neuron to the i^{th}neuron

After network creation and the weights initialisation, the feedforward algorithm will be implemented according to formula 2.1 as shown in the *Pseudo Algorithm 2 Feedforward Propagation;*

Pseudo Algorithm 2 Feedforward Propagation

```
procedure:feedforward

    for each inputVector in Dataset do

        for each layer in in network do

            for each neuron in layer do
                weightVector <- getWeights OfNeuron(neuron)
                sum=inner_product(inputVector , weightVector )
                sum=function(sum)

```

Figure 4.2 Pseudo code of feed forward propagation; for each neurons' weight vectors in layers is multiplied by input vector in dataset.

4.5.2 Backpropagation Algorithm

For the next step of the project, backpropagation has been chosen as an optimisation algorithm. As discussed in Chapter 2, this algorithm works by taking derivative of error function (Eq 2.4) with respect to network weight and then those weight values are updated depending on the protocol that is explained below. Pseudo Algorithm 3 Backpropagation

Pseudo Algorithm 3 Backpropagation

```
procedure:backpropagation
    Begin initialise realOutputVector , i<-0;

    Since backpropagation function is called right after feedforward, the
    output of network can be  called by getOutput() function directly
    outputVec <- getOutput();

    for each element in realOutputVec do
        i <- i+1
        deltaVector[i]<- (realOutputVec[i]- outputVec[i]) * previousNeuronValues.
                         * (1- previousNeuronValues);

    calculateWeightChanging(deltaVector);
```

Figure 4.3 Pseudo code of Backpropagation; after obtaining output vector of network, it is ssubstracted with real output vector that comes from dataset. After that is multiplied by previous neuron value and 1-previous neuron which equivalent to derivative of neuron net value (Equation 2.16)

4.5.2.1 Updating Weights - Training Protocols

As can be seen in ***Pseudo Algorithm 3*** *Backpropagation* line 14, the function *"calculateWeightChanging*" is called to store weight changes after the delta vector of the network is calculated. The pseudo code of the *calculateWeightChanging function* as follow;

Pseudo Algorithm 4 Weight changes calculations

```
1:procedure: calculateWeightChanging
2:      Begin initialise deltaVector ,previousNeuronValues , i<-0,k<-0;
3:
4:
5:       for each neuron  in layer do
6:              i <- i+1
7:              k <- 0;
8:
9:              for each weightDelta  in neuron  do
10:                       k <- k+1
11:
12:                       Weight changing is implemented (Eq 2.10)
13:                       weightDelta  = learningRate* previousNeuronValues [k]*deltaVector[i] +
                                                    momentum* weightDelta
14:
```

Figure 4.4 Pseudo code of calculateWeightChanging function; After calculation delta values in bakpropagation function, it is passed to calculateWeightChange function and in every neuron weightDelta value is updated by multiplying learning rate, previous neuton values, delta values, and momentum (Equation 2.8).

On line 13, there is an addition term *momentum (α)* which differs from Eq 2.10. The momentum parameter determines the quantity value to add weights by using previous and current weight values (Eq. 4.3).

$$\Delta w_i = \eta\ (y_j - o_j) f'(net_j)\ x_i + \ \alpha \Delta w_i \qquad (4.2)$$

$$w_i = \ w_i + \ \Delta w_i \qquad (4.3)$$

The advantages of momentum is that due to Δw values depending on learning rate and derivative of error function (Eq. 2.10), in the case of derivative of error tending to be smaller value, there should be a parameter to make descending step bigger. By making steps bigger the error minima can be reached in fewer steps. The difference between using and not using momentum can be seen in Figure 4.2;

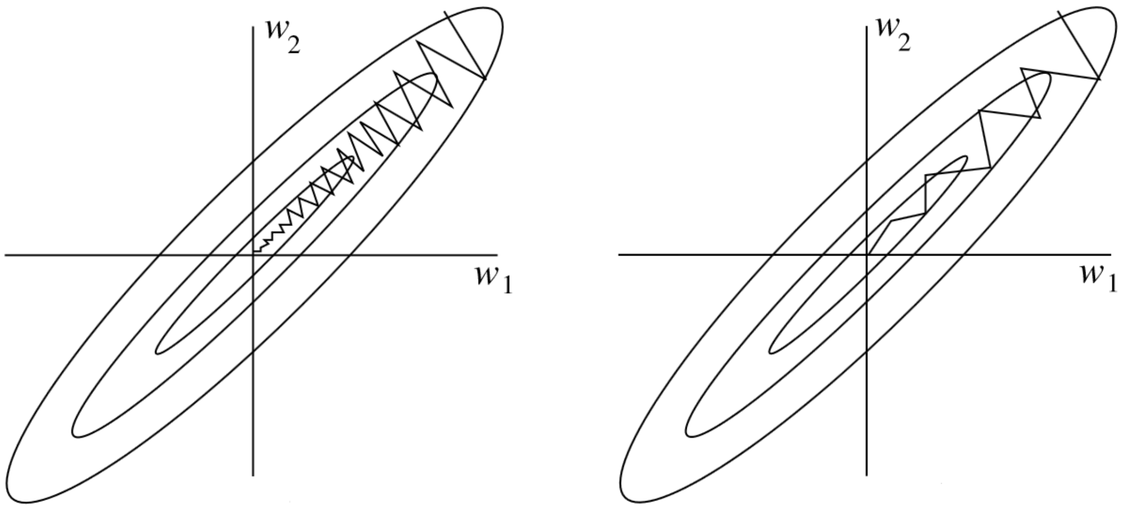


Figure 4.2: Updating weights by using momentum (right) and not using momentum (left); as it is seen in Figure 2.10; the desired point (the centre of contour plot in this case) is getting be more closer in each weight update. Each arrow above represents a step toward to centre and in the case of not using momentum; due to gradient descent is decreasing while approaching to centre, the steps are getting be smaller which cause to more calculation burden that takes more time network to converge. On the other hand when momentum is used, even though gradient descent is a small number value, momentum balances the step length and the network is converged faster with less calculation

As can be seen, the steps of descending are getting smaller and smaller (left) because the derivative of error function tends to be smaller and that causes more iteration, however by using momentum (right) the steps become more stable and the minimum point of error function is reached faster than without using the momentum method.

Considering the advantages of momentum, it is used in the project weight updating protocols which will be implemented every 10 samples in the way as that was done by Khan et al., study (2001). In regarding to weight update regularity, there couple of different update protocol and considering to future usign of Cunernet Library, the main two of those protocol (batch, stochastic) also will be added to the project software and those will be able to be used as optional.

Chapter 5

Implementation, Testing and Evaluation

This chapter will describe the training implementation of the neural network, the Cunernet software implementation and the test results, and finally an evaluation of the work will be presented.

5.1 Training Implementation

The project network in training/testing procedures is restricted to the single layer neural network as it has been done in Khan et al., (2001) study. The dataset is read from the given dataset and input layer neuron numbers is set to 96 due to the each sample has 96 genes values and also 4 output neuron is created in context of the output vectors have 4 elements (Figure 5.1)

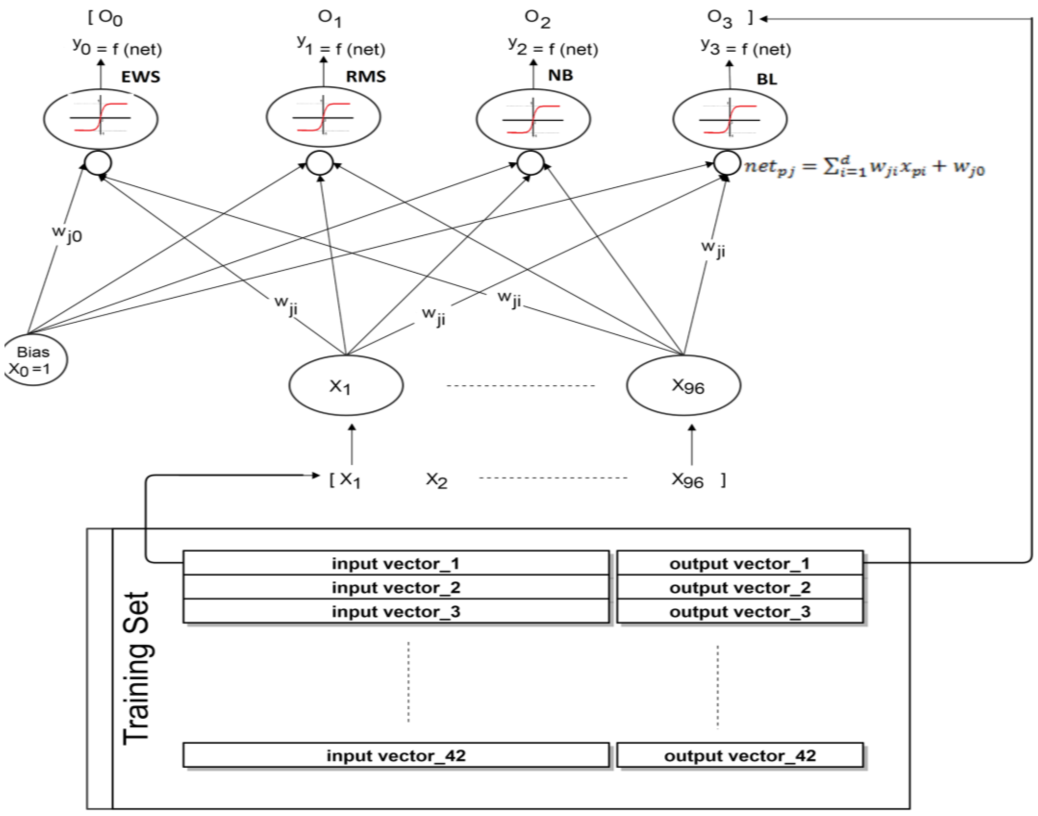
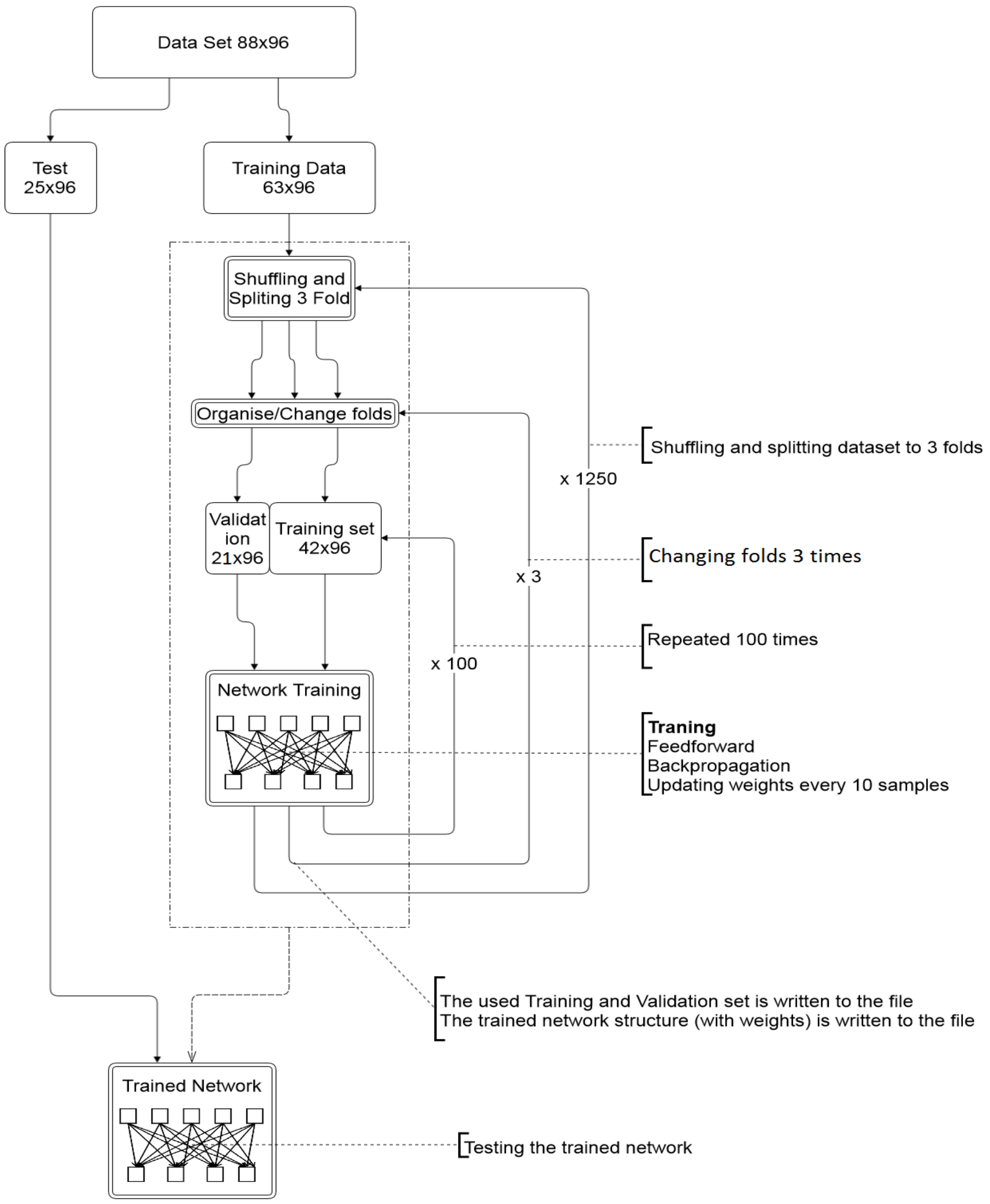


Figure 5.1 The used network structure; training set consist of 42 samples and each sample has input (96) and output (4) vector. While input vector has 96 elements correpond to genes, the output vector has 4 elements corresponds to category of cancer. (presence of category is represented by 1 , 0 otherwise in output vector). In training procedures, input vector of sample is given to the network and the network output vector (y) is obtained. Thereafter obtained output vector and the real output vector, mentioned above, is substracted and through the calculations mentioned in *Chapter 2*, backpropagation is implemented. These procedures is repeated untill all samples is used.

After the network is built, training procedures is started by reserving 63 over 88 samples to train network while the remaining 25 samples are used to test network correctness. In the beginning of training network, 63 samples are shuffled and in context of cross validation it is split into three-fold. While any two of three folds (42 samples) is labelled as training set, the other 21 samples set as validation. The training set is given to the network through the feedforward, backpropagation and every 10 samples the weights of network is updated (Eq. 4.4) as it explained in *Chapter 4*. These procedures is implemented 100 times by using same training set and at the end of 100 times repetition the network is saved to a file with its weight to be used in testing. This process is repeated 3 times and each time different 21 samples are used as validation. After finishing 3 folds using three different network is collected and 1 iteration is completed. After finishing 1 iteration, whole procedures is started again with shuffling whole dataset and splitting to 3 folds again. The training of network is completed after 1250 iteration as it has been done in Khan et al., (2001) study. The performed tasks by each block of flowchart can be seen in Figure 5.2



Data Set 88x96
Test 25x96
Training Data 63x96
Shuffling and Spliting 3 Fold
Organise/Change folds
Validat ion 21x96
Training set 42x96
Network Training
x 1250
x 3
x 100
Shuffling and splitting dataset to 3 folds
Changing folds 3 times
Repeated 100 times
Traning
Feedforward
Backpropagation
Updating weights every 10 samples
The used Training and Validation set is written to the file
The trained network structure (with weights) is written to the file
Trained Network
Testing the trained network

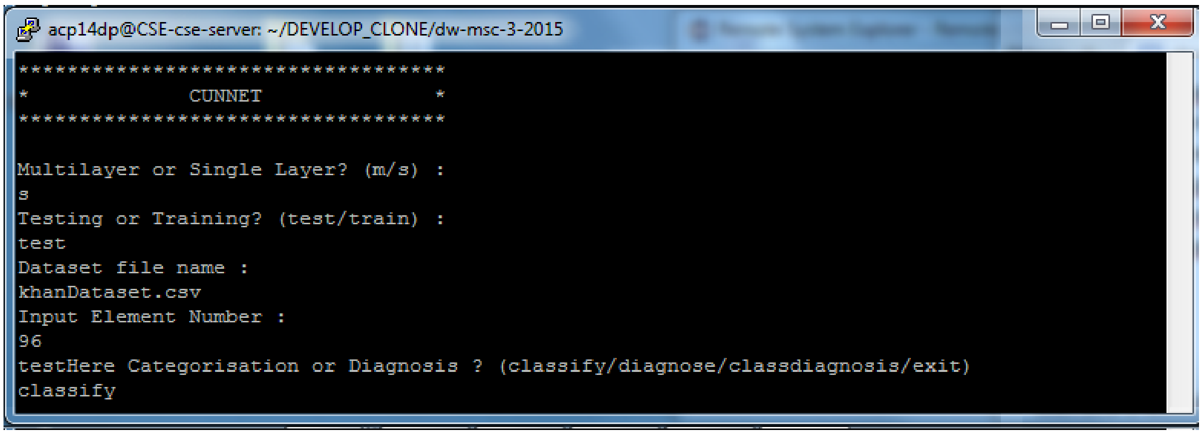
Figure 5.2 Training Procedures; after obtaining 63x96 dataset, it has been split to 3 folds and while 2 of 3 is used as training 100 times, the other 1 fold remained as validation. Thereafter the validation fold is changed and these procedures is repeated three times with different validation sets in each. After completion 3 times changing, 1 iteration is accomplished and the all training network stage is finished after 1250 iterations as it has been done in Khan et al., (2001).

5.2 Software Implementation

The Library Cunernet is written by using C++11 language in object oriented programming paradigm. It can be downloaded from the Bitbucket project page and easy to compile with commands that are written in the readme file.

After compiling and launching software, the question "*Training or Testing?"* is asked that determines whether it is going to be run to train or test network.

If *Testing* is chosen, a dataset name should be given with gene number in it. After passing proper information, the question for classification or diagnosis is asked. In this step there is a third option (*classdiagnosis*) is added as an extra to the Khan et al.,(2001) methods (Figure 5.3). By choosing "classify", the saved networks in training are testes on samples one by one. When "diagnose" is chosen it asks the training sample number to calculate distance threshold, that explained in *5.4 Testing Network Correctness and Evaluation,* after giving training sample number it starts calculate distances and end of it diagnosis result is given to the command panel (Khan et al., 2001), When "classdiagnosis" the testing procedures is typed it starts trying each saved network with all samples in given dataset and end of trying all networks the best result is given to the panel..

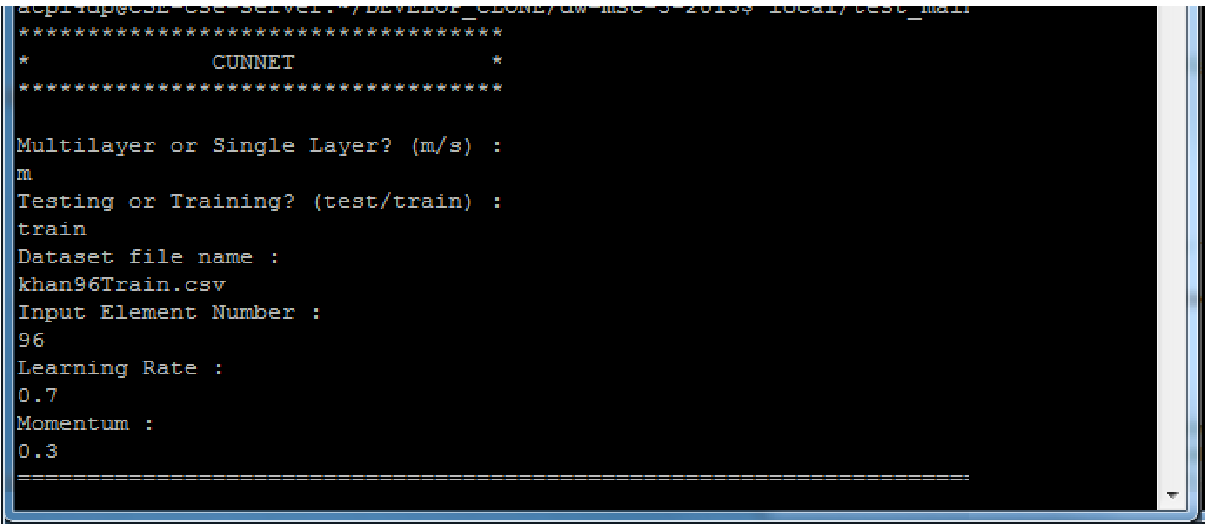


```
acp14dp@CSE-cse-server: ~/DEVELOP_CLONE/dw-msc-3-2015
***********************************
*            CUNNET              *
***********************************

Multilayer or Single Layer? (m/s) :
s
Testing or Training? (test/train) :
test
Dataset file name :
khanDataset.csv
Input Element Number :
96
testHere Categorisation or Diagnosis ? (classify/diagnose/classdiagnosis/exit)
classify
```

Figure 5.3 Cunernet library testing procedures starts. By giving proper information classification started. Also diagnosise and classdiagnosis could have been chosen.

If the *Training* is chosen, the dataset name with its gene number and also learning rate momentum values information is asked (Figure 5.4).



```
***********************************
*            CUNNET              *
***********************************

Multilayer or Single Layer? (m/s) :
m
Testing or Training? (test/train) :
train
Dataset file name :
khan96Train.csv
Input Element Number :
96
Learning Rate :
0.7
Momentum :
0.3
==================================================================
```

Figure 5.4 Cunernet library training procedures starts by passign proper information.

After giving that information, the iteration number, and the average sum square error value and the correctness percentage of the training datasets are shown to the user, as can be seen in the panel above. During this training process, the procedures that are explained in Figure 5.2 are implemented. As is shown in the flowchart, every training and validation set used are recorded to the file (train_validation.csv") and also before changing folds (*see* Figure 5.2) the updated network structures are written to files and named as 'model'iterationNo'.csv' (for example, 'model44.csv, 'model45.csv' and so on) in each fold-changing step. At the end of the training procedure, the 3750-model network is written to files and in testing implementation those files are reloaded to test the network correctness with the remaining 25 samples as was explained in Section 5.4 in detail (the files can be found in the Bitbucket project data folder).

5.3 Testing

In this section, first the over-training control procedures, unit testing and profiling code operations will be defined. Then the main 25 samples used for determining the trained network correctness testing will be explained, with the results.

5.3.1 Over-training control

In order to check for the existence of over-training in the calibration process, a sample low dimensional dataset was created from the 63x96 dataset used and it was given to the network. During the creation of the low dimensional dataset, various functions which are explained below were used from the *Neural Network toolbox Gene Expression Analysis* example (MATLAB Neural Network Toolbox Release,R2014a).

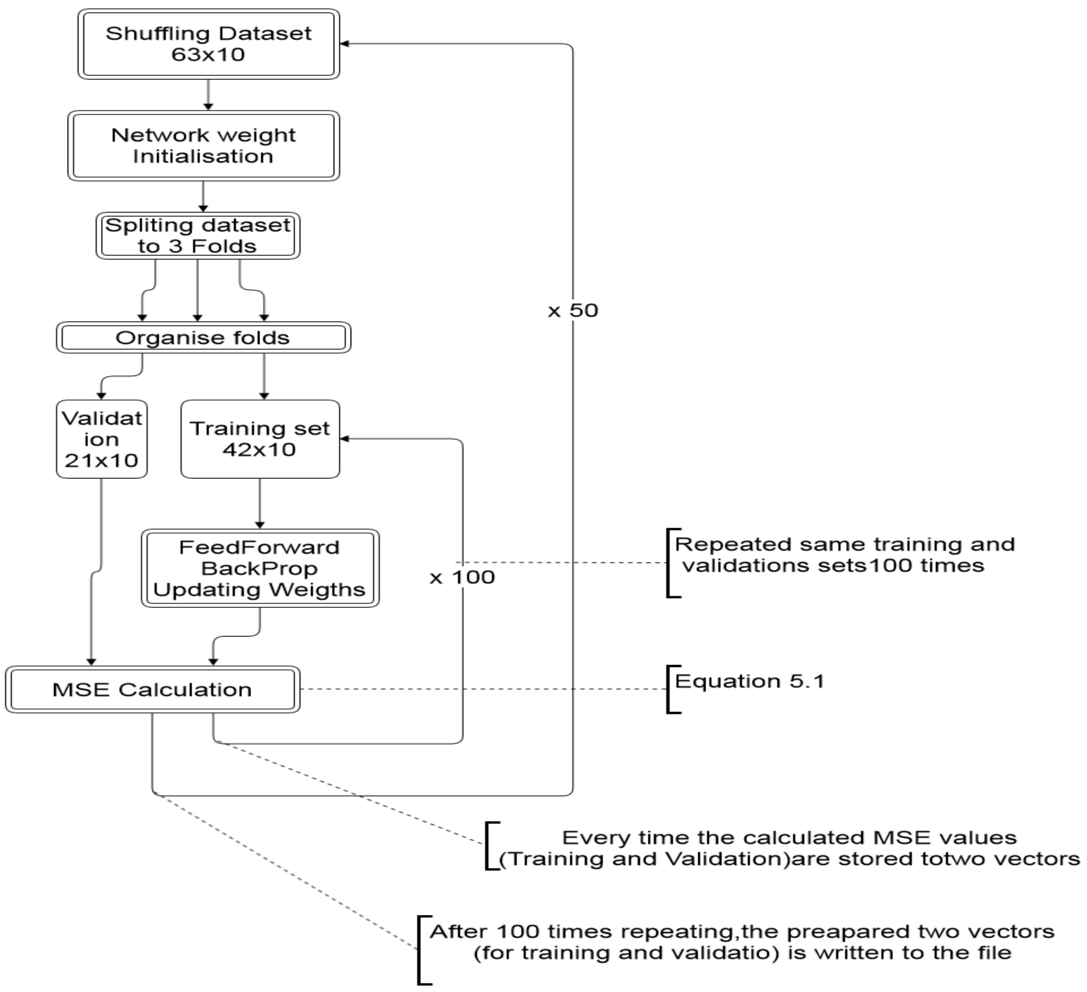
- *Genevarfilter: Filtering genes with small profile variance in dataset*
- *Geneentropyfilter: Removing genes with low entropy expression values in dataset*
- *Genelowvalfilter: Removing gene profiles with low absolute values*
- *Generangefilter: Removing gene profiles with small profile ranges*
- *Processpca: Reducing dimension of data by using orthogonal transformation.*

After filtering genes on Matlab, the processpca was used as a final function to reduce the dimensionality of the rest of the data and a sample dataset consisting of ten components (63x10) was obtained and used in the network calibration procedure.

At the beginning of over-training control, the network is structured as 10 input and 4 output neurons and the weights are initialised. Then the prepared dataset (63x10) is split 3 folds and two of them used as training set in calibration operations feedforward ,backpropagation and updating weights. The remained one fold is kept as validation and only feedforward propagation is used on this validation set. This operations are implemented 100 times and each time MSE (i.e. mean square error) is calculated (Eq. 5.1) for both the training and validation set. After the calculated MSE values are stored in two vectors separately, at the end of 100 iterations these vectors are written as the errLog.file file and the over-training procedures are relaunched by initialising the network with the same weights and shuffling the data (see Figure 5.5).

$$E = \frac{1}{2N_p}\sum_{p=1}^{N_p} \sum_{i} (y_i^p - o_i^p)^2 \qquad (5.1)$$

N_p : number of sample in dataset



Shuffling Dataset 63x10
Network weight Initialisation
Spliting dataset to 3 Folds
Organise folds
Validation 21x10
Training set 42x10
FeedForward BackProp Updating Weigths
MSE Calculation
x 50
x 100
Repeated same training and validations sets100 times
Equation 5.1
Every time the calculated MSE values (Training and Validation)are stored totwo vectors
After 100 times repeating,the preapared two vectors (for training and validatio) is written to the file

Figure 5.5: Over-Training Testing;The network is initialised with chosen values between [r,-r], then the dataset is split to 3 folds and any two of them is used as training set. The network is trained by using the chosen training set and thereafter MSE values are calculated for both the chosen trainign , validation sets and the calculated MSE values are stored to two different vector seperately. By using same training and validation sets, same procedure is repeated 100 times and each time 2 MSE values (for validation and training) are added to the two vectors seperately. At the end of 100 repetation, 2 vectors with a hundred MSE values (1 for training 1 for validation) are obtained and written to a file to drawn later on. These vectors with 100 MSE elements will be shown in Figure 5.6 as one line for each vector (purple for training and grey for validation). By doing that 1 main iteration is completed and after 2 vectors are cleared, the weight are re loadeded to network and second main iteration procedures are re-run by shuffling data uniquely. This main iteration is repeated 50 times to obtain enough statistical informatino of network over-training. At the end of this testing as 50 for each sets we got 100 vectors in total.

In order to obtain enough statistical information, the procedures above is repeated 50 times and by doing that 100 vectors (50 for training , 50 for validation sets) are obtained in the file errLog.csv. These 100 vectors ,each corerespond to the one line, are used to check whether over training exist or not. By using a python script (plotTrainAndValidate.py) that can be found in scripts folder on Bitbucket project, the error values are plotted on a line graph (Figure 5.6)

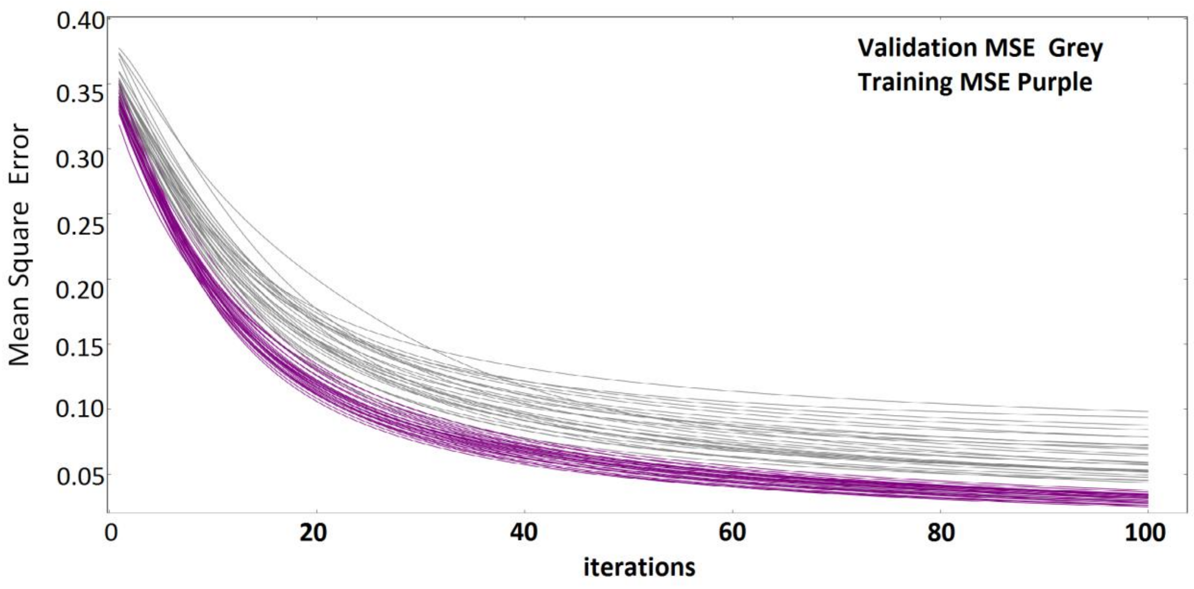


Figure 5.6 MSE calculations; By using same training and validation sets, the MSE values of both (for training and validation) is calculated 100 times and stored to two vectors.At the end of this repetiton, two vectors that have 100 MSE values in each is created. This processes are repeated 50 times and 100 different vectors (50 for training(purple),50 for validation(grey)) are collected and plotted.

As can be seen in Figure 5.6, whilst the errors of the training set (purple) decreased through the 100 repetitions, the validation errors (grey) also decreased similarly. Because there was no increase in validation errors, it can be said that there is no sign of over-training in the network training.

5.3.2 Unit Testing

Unit testing is a method to check procedures of software whether it is suitable for use or not. Unit test is implemented in order to test the correctness of the present study software training procedures by creating datasets and training network manually.

The dataset was structured as four input elements and two output elements which led to four-neuron input and two-neuron output layers. To start testing, the input vectors are given to the network one by one and the training procedures in Figure 5.2 are followed. During the training of the network, both network output elements and real output elements are written to a file in each iteration to monitor the behaviour of the network.

Iteration 150

Fold 3			
Train Set			
Network Output		Dataset Real Output	
0.000125	0.999875	0	1
0.98782	0.012209	1	0
0.995976	0.004032	1	0
0.996673	0.003331	1	0
0.996673	0.003331	1	0
0.99812	0.001877	1	0
0.000174	0.999826	0	1
0.005957	0.994041	0	1
0.971165	0.028866	1	0
0.995976	0.004032	1	0
0.94422	0.055724	1	0
0.997726	0.002272	1	0
0.283181	0.716934	0	1
0.992811	0.007204	1	0
Validate Set			
Network Output		Real Output	
0.933273	0.066712	1	0
0.99812	0.001877	1	0
0.002515	0.997477	0	1
0.00073	0.999271	0	1
0.96682	0.033206	1	0
0.996673	0.003331	1	0

Table 5.1:Unit Test Results; The network outputs are converging to the dataset real output values by getting be more and more close to then (1 or 0).

As the example in Table 5.1 shows, the output vectors of the network converged to the real output values for both the training and the validation sets, which shows that training the network worked as required and is therefore appropriate for use in real examples.

5.4 Testing Network Correctness and Evaluation

After these training procedures are done, the obtained network models are reloaded from the files and tested on 25 test samples which consist of 6 NB, 6 EWS, 3 BL,5 RMS and 5 different cancer subtype as noise information. It is worth to remain that, these 25 samples is seperated from 88 samples dataset at the beginning of training procedures and they are never involved training procedures which means completely new for the network.

Ewing Family of tumours EWS	Rhabdomyosarcoma RMS	Burkitt lymphomas BL	Neuroblastoma NB	Noise
6 Samples	5 Samples	3 Samples	6 Samples	5 Samples

Table 5.2: The test samples cancer category distribution in 25 samples

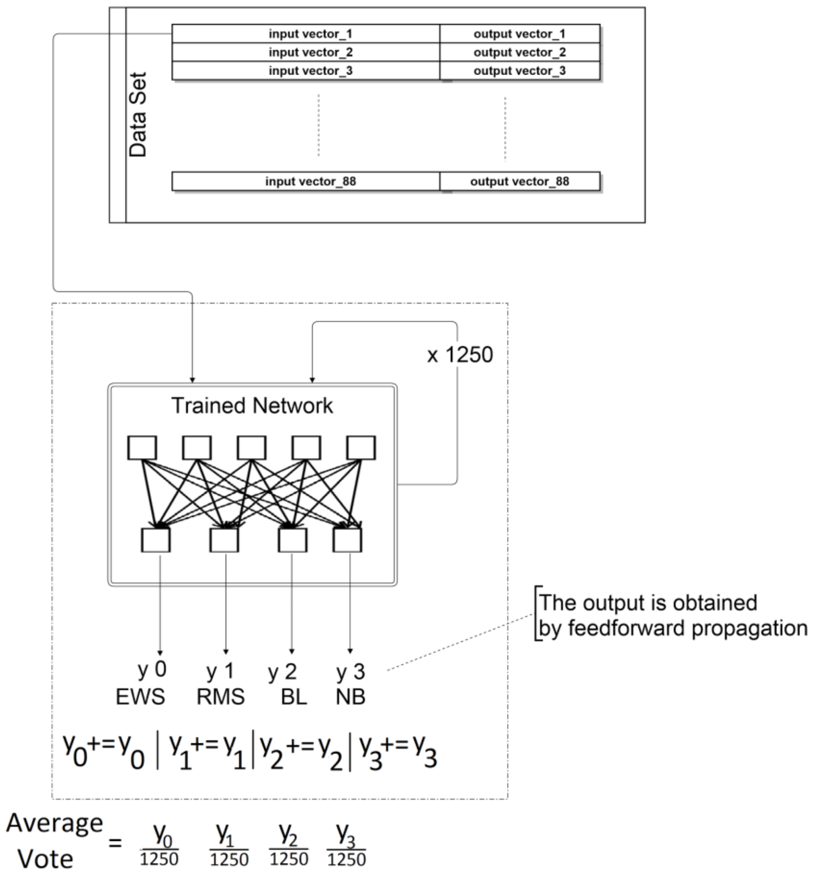
According to 4 different cancer category is included in the test dataset, the output of each sample consist of 4 elements and while category of cancer is denoted by 1 the rest of three elements is denoted by 0. For instance if the first element of output is 1 it shows that this sample is EWS type.

As it mentioned above, testing has 3 options; first two is *classification* and *diagnosis* methods which have been used in Khan et al., (2001) study and *classDiagnosis* method that is used in the presented study as an extra. The procedures in classification and dianosis are as follow;

5.4.1 Classification

As in the Khan et al., (2001) did, a term average vote is used in the classification process. To find average votes, firstly a network over 1250 saved network is reloaded and one sample is taken form dataset. Feedforward propagation is implemented by using the taken sample and the network output is gathered. Thereafter a second network over 1250 is reloaded and by using same sample, feedforward is implemented on the second reloaded network. The gathered second network output is accumulated with the first gathered network output. After reloading 1250 network with accumulation network outputs, the average vote for the first taken sample is obtained and all procedures are implemented for all samples one by one to obtain all averate votes of samples.

After obtaining all samples average votes, the classification is done by checking maximum value number index in the average vote vector. Since the network has 4 output neurons, which correspond to 4 cancer category (EWS,RMS,BL,NB), the average vote vector has 4 element as well. The real/ target output vector, which are in the dataset, shows the presence of cancer with 1 and 0 otherwise. For instance a vector with elements 1,0,0,0 indicates that the sample has EWS category cancer. Therefore if maximum value in obtained average output vector is in the first index, it is classified as EWS, second for RMS, third for BL and latly fourth for NB.



Data Set
input vector_1
input vector_2
input vector_3
input vector_88
output vector_1
output vector_2
output vector_3
output vector_88
x 1250
Trained Network
The output is obtained by feedforward propagation
y 0 EWS
y 1 RMS
y 2 BL
y 3 NB
$y_0 += y_0 \mid y_1 += y_1 \mid y_2 += y_2 \mid y_3 += y_3$
Average Vote = $\frac{y_0}{1250}$ $\frac{y_1}{1250}$ $\frac{y_2}{1250}$ $\frac{y_3}{1250}$

Figure5.7: Classification procedures; a sampe is given to the saved 1250 network by feedforward propagation. Thereafter the outputs are accumulated for each network and after using all saved networks, the accumulated outputs are divided by the network number to find average vote of the used sample. These procedures are done for all samples and according to the maximum element index of average vote vector it is classified.

After finishing the classification of the Cunernet Library it can be seen that all samples are classified with 100% accuracy that means all samples in dataset of 88 samples is classified correctly (*see* Table 5.3) as was done by Khan et al., (2001) study. Also it should be noted that, the "noise" samples are classified as noise in the Cunernet library classification. The classification of noise is done according to the weight of maximum number in the four element vector. If the maximum number has a less than 98% weight then it is labelled as noise. (The 98% threshold was decided after several trials) At the end of trials it is seen that 5 over 5 noise samples is classified correctly while Khan et al., (2001) did not classify none of them.

		Real Output	Network Output	Classification	Network Outputs EWS	RMS	NB	BL
Sample	1	0	0	Matched	0.999999	9.26E-06	1.28E-06	7.54E-05
Sample	2	0	0	Matched	0.999994	8.24E-06	3.81E-05	2.52E-05
Sample	3	0	0	Matched	0.999999	7.03E-05	1.02E-07	4.93E-07
Sample	4	0	0	Matched	0.999996	0.0001377	7.96E-09	1.39E-07
Sample	5	0	0	Matched	1	7.50E-07	1.03E-07	6.76E-06
Sample	6	0	0	Matched	1	1.86E-07	3.02E-07	1.21E-05
Sample	7	0	0	Matched	1	9.71E-06	6.16E-08	3.63E-07
Sample	8	0	0	Matched	1	4.71E-08	1.35E-07	1.29E-06
Sample	9	0	0	Matched	1	2.59E-09	1.03E-08	1.69E-06
Sample	10	0	0	Matched	0.999428	0.0001402	5.30E-12	2.25E-08

Sample	11	0	0	Matched	1	1.59E-07	1.95E-06	6.41E-05
Sample	12	0	0	Matched	1	1.85E-07	5.96E-06	2.39E-06
Sample	13	0	0	Matched	1	1.59E-06	2.70E-06	3.65E-07
Sample	14	0	0	Matched	1	6.82E-07	3.25E-07	1.32E-05
Sample	15	0	0	Matched	0.999944	6.85E-06	0.000435	2.49E-05
Sample	16	0	0	Matched	0.999018	0.0001302	0.0003298	0.0002025
Sample	17	0	0	Matched	0.99931	0.0005445	0.0002998	0.000369
Sample	18	0	0	Matched	0.999831	0.0001134	8.36E-05	0.0001262
Sample	19	0	0	Matched	0.999963	6.51E-05	3.24E-05	6.66E-05
Sample	20	0	0	Matched	1	4.42E-06	2.92E-06	4.00E-06
Sample	21	0	0	Matched	0.99952	3.30E-05	0.0013769	6.46E-05
Sample	22	0	0	Matched	0.999618	5.54E-05	0.0009757	0.0004503
Sample	23	0	0	Matched	0.998753	0.0006452	0.0002926	9.89E-05
Sample	24	3	3	Matched	9.77E-05	2.73E-05	0.0001647	0.999963
Sample	25	3	3	Matched	0.000262342	3.13E-05	0.0004996	0.999769
Sample	26	3	3	Matched	4.94E-05	3.01E-05	5.85E-05	0.99998
Sample	27	3	3	Matched	8.01E-05	1.38E-05	4.81E-05	0.999955
Sample	28	3	3	Matched	0.000123608	0.0008356	0.0007692	0.999382
Sample	29	3	3	Matched	0.000281674	0.0002109	0.0003967	0.999878
Sample	30	3	3	Matched	0.000160402	0.0003014	0.0001588	0.999756
Sample	31	3	3	Matched	0.000195608	0.0002678	0.0006139	0.999087
Sample	32	2	2	Matched	1.09E-06	5.90E-05	0.993575	2.29E-06
Sample	33	2	2	Matched	6.07E-07	8.10E-05	0.993075	0.0001272
Sample	34	2	2	Matched	8.80E-08	0.0048147	0.990776	9.28E-08
Sample	35	2	2	Matched	0.00075952	3.95E-05	0.993535	0.0005605
Sample	36	2	2	Matched	7.76E-05	8.42E-05	0.993505	0.0002416
Sample	37	2	2	Matched	5.10E-05	5.32E-05	0.991245	0.0002521
Sample	38	2	2	Matched	0.000266827	1.25E-05	0.993562	0.0001695
Sample	39	2	2	Matched	0.000257855	2.19E-06	0.993594	8.66E-05
Sample	40	2	2	Matched	8.06E-05	0.0002301	0.992751	0.0002164
Sample	41	2	2	Matched	0.00117653	4.54E-05	0.992208	4.89E-05
Sample	42	2	2	Matched	0.000538316	1.27E-05	0.993155	0.0001874
Sample	43	2	2	Matched	4.59E-06	0.0004662	0.993459	6.50E-06
Sample	44	1	1	Matched	6.35E-09	0.999997	1.67E-05	1.27E-07
Sample	45	1	1	Matched	8.07E-05	0.999623	1.97E-05	1.12E-06
Sample	46	1	1	Matched	1.19E-05	0.999968	3.10E-06	5.61E-07
Sample	47	1	1	Matched	1.87E-05	0.999583	2.89E-06	7.95E-08
Sample	48	1	1	Matched	5.41E-06	0.999252	0.0008625	2.60E-06
Sample	49	1	1	Matched	8.44E-06	0.998939	0.0013327	1.35E-05
Sample	50	1	1	Matched	1.12E-05	0.999456	0.0008092	6.35E-06
Sample	51	1	1	Matched	0.000749966	0.999184	0.0006292	4.45E-06
Sample	52	1	1	Matched	5.92E-06	0.999277	0.0006634	1.06E-06
Sample	53	1	1	Matched	0.000285102	0.999078	4.92E-05	7.15E-07
Sample	54	1	1	Matched	7.84E-08	0.999994	9.67E-08	3.37E-06
Sample	55	1	1	Matched	3.33E-05	0.999993	6.92E-09	1.23E-08
Sample	56	1	1	Matched	0.000436614	0.999953	1.11E-06	4.02E-06
Sample	57	1	1	Matched	1.07E-06	0.999973	2.64E-07	7.93E-10
Sample	58	1	1	Matched	9.77E-05	1	7.77E-07	2.29E-08
Sample	59	1	1	Matched	5.74E-05	0.99992	1.15E-08	4.90E-09
Sample	60	1	1	Matched	1.84E-07	0.999997	1.67E-08	5.25E-08
Sample	61	1	1	Matched	0.000619259	0.999953	8.29E-07	7.31E-08
Sample	62	1	1	Matched	0.00059364	0.99977	2.54E-06	3.92E-09
Sample	63	1	1	Matched	7.39E-11	1	6.73E-11	5.71E-11
Test	1	2	2	Matched	0.0120638	2.28E-06	0.993532	0.0004412
Test	2	0	0	Matched	0.999628	4.51E-05	0.0001844	4.66E-05
Test	3	-1	-1	Matched	0.549998	0.0031857	0.0009182	0.0072279
Test	4	1	1	Matched	4.44E-10	1	9.80E-08	5.69E-08
Test	5	-1	-1	Matched	0.0851901	0.0038741	0.0185478	0.0755261
Test	6	0	0	Matched	1	1.31E-07	2.68E-06	1.55E-05
Test	7	3	3	Matched	0.000272146	2.50E-05	0.0001219	0.999985
Test	8	2	2	Matched	0.000172695	3.73E-06	0.993592	3.12E-05
Test	9	-1	-1	Matched	0.640664	0.150756	3.31E-05	0.0080591
Test	10	1	1	Matched	5.37E-09	0.999971	1.64E-10	7.10E-08
Test	11	-1	-1	Matched	0.454711	0.0033484	0.0102645	0.0056509

Test	12	0	0	Matched	1	2.13E-07	6.02E-06	1.33E-07
Test	13	-1	-1	Matched	0.768471	0.938202	1.77E-06	3.13E-05
Test	14	2	2	Matched	1.09E-05	2.15E-10	0.9936	0.0026604
Test	15	3	3	Matched	6.52E-05	3.29E-08	1.46E-06	0.999999
Test	16	2	2	Matched	8.81E-06	1.42E-09	0.9936	0.0001355
Test	17	1	1	Matched	9.75E-07	0.999978	4.06E-06	1.32E-07
Test	18	3	3	Matched	0.00299425	0.0005032	0.0012095	0.997766
Test	19	0	0	Matched	1	3.49E-08	1.24E-06	7.85E-06
Test	20	0	0	Matched	0.849812	9.23E-05	9.05E-08	0.0006389
Test	21	0	0	Matched	0.999386	0.0001503	6.09E-12	2.45E-08
Test	22	1	1	Matched	1.30E-07	0.999999	5.60E-07	2.81E-07
Test	23	2	2	Matched	6.58E-05	8.30E-08	0.986638	2.91E-05
Test	24	1	1	Matched	3.16E-06	0.999999	2.78E-08	6.08E-08
Test	25	2	2	Matched	3.52E-06	1.80E-11	0.9936	0.0003068

```
##################################
Classification is done with 100%  accuracy
##############################
```

Table 5.3; Classification Result; Real output column represent the real diagnosis and 0 for EWS, 1 for RMS, 2 for NB , 3 for BL and -1 for noise samples. Network output column represent the classification result of the Cunernet.

5.4.2 Diagnostic Classification

In diagnostic classification, a term "distance" is used as was done by Khan et al., (2001). First, a sample from training dataset is given to the 3750 saved network and this sample's output is obtained by feedforward propagation. Then the distance (Eq. 5.2) of this sample is calculated and accumulated as is in Figure 5.8. This calculation/accumulation was implemented all training samples and at the end of it, the samples are grouped by classification result and 95% of the total distance was taken as a threshold value. Then the test samples' distance were calculated on all 3750 saved networks and if their distance is less than calculated threshold value, it is diagnosed, otherwise it is labelled as noise.

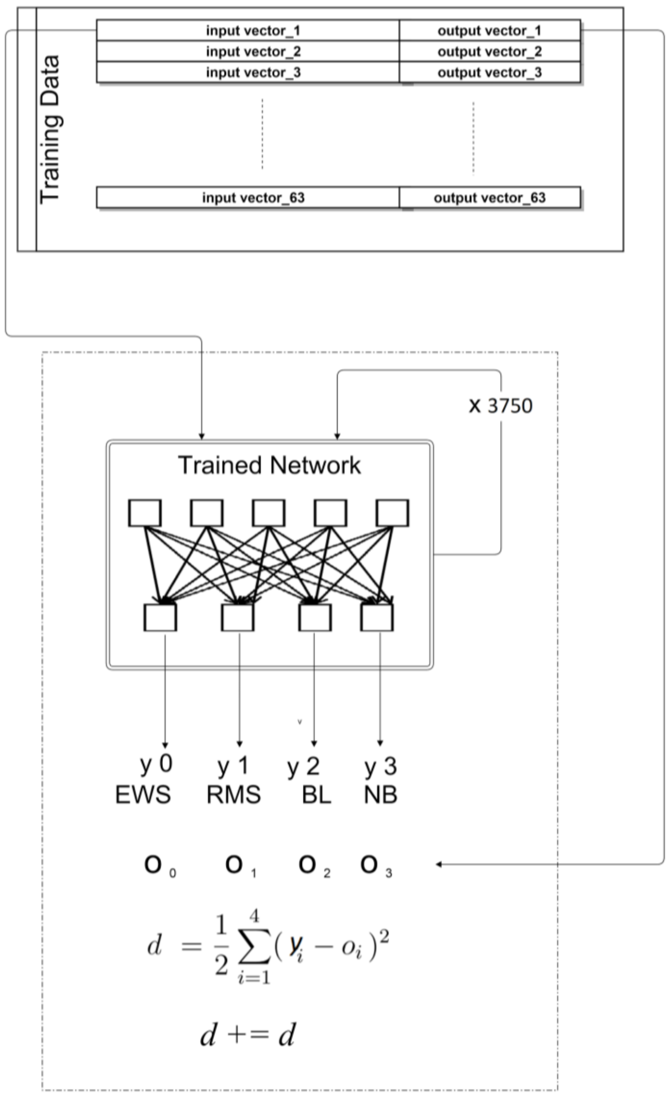


Figure5.8 Diagnosis procedures; Each samples' distance is calculated. 95% of training samples distances are calculated as an treshold values for each cancer category. Thereafter each test sample's distance calculated and according to classification type treshold values it is checked wheter less or more than. In the case of being less than treshold value (95% of sum of training samples distances) it is diagnosed otherwise set as a noise.

$$\boldsymbol{d = 1/2 \sum_{i=1}^{4} (y_i - o_i)^2} \quad \mathbf{5.2}$$

At the end of these diagnosing procedures, our Cunernet Library diagnosed all samples with a 96% accuracy, the same result as that achieved by Khan et al., (2001) (see Appendix Table 1).

5.4.3 ClassDiagnosis:

In this method, one reloaded network is given by all samples at once and the each saved network correctness is considered separately. The steps of this method are followed;

- Firstly a network over 3750 is reloaded.
- A sample is given to the reloaded network and feedforward is implemented

- The network output of this sample is obtained and it is classified according to the highest number in the obtained vector.
- To diagnose this obtained vector. The highest value weight in all four element is checked
- If the highest value weight is ore than 98% than is it diagnosed according to the highes value index the diagnosis result is checked with real diagnosis.
- Thereafter by using same reloaded network, second sample is given to the network and same procedures are done.
- At the end of all samples, the reloaded network is correctness accuracy is calculated according to correct diagnosed sample number.
- Than second network is reloaded and all samples is given to the second network again.
- At the end of calculating all network correctness accuracy, the network which has a highest accuracy is chosen to be shown.

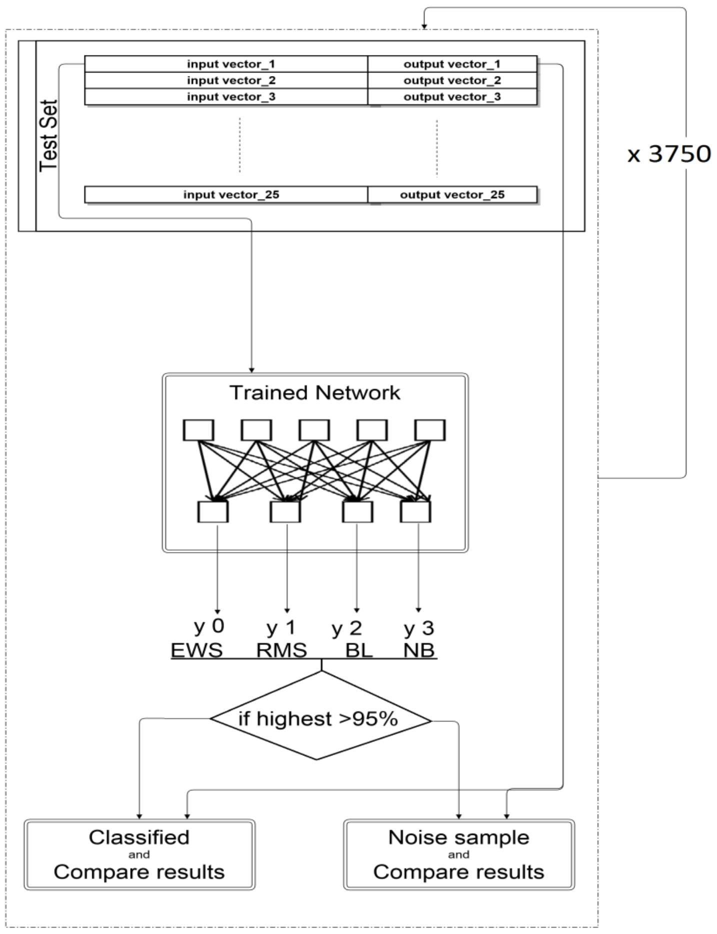


Figure 5.9: ClassDiagnosis Procedures; The saved network are taken 25 samples one by one. If the output of network has a value which has 98% or more weight in all four values, it is diagnosed and compared to real output value. If none of the four values has 98% weight, it is diagnosed as noise sample and compared to real output value. Overall all matches are sum and divided by 25 to gain accuracy percentage of trained/reloaded network.

As can be seen in Table 5.4, the results of comparison show that the network has classified 20 samples with 100% accuracy and 24 samples of the 25 have been diagnosed correctly. Only “Test sample 20" was diagnosed as noise, whereas it was a EWS sample. Apart from that, all 24 samples were diagnosed correctly which is identical to Khan et al., study (2001) results. The comparison table between the presented study and that of Khan et al., (2001) study can be seen in Table 5.5.

Samples	Network Output Vector				Classification	Diagnosis	Real Diagnosis
	EWS	RMS	NB	BL			
TEST.1	0.019555	9.63E-05	0.999209	0.00221758	NB	NB	NB-C
TEST.2	0.980755	0.000515	0.00672988	0.00042237	EWS	EWS	EWS-C
TEST.3	0.589736	0.017952	0.0174805	0.0473966	EWS	-	OsteosarcomaC
TEST.4	9.23E-09	1	1.31E-06	1.32E-06	RMS	RMS	RMS-T
TEST.5	0.229616	0.023906	0.0380616	0.0629722	EWS	-	Sarcoma
TEST.6	0.999853	2.99E-05	6.86E-05	0.00017739	EWS	EWS	EWS-T
TEST.7	0.00636	0.001244	0.00285877	0.99905	BL	BL	BL-C
TEST.8	0.00155	0.000152	0.999619	0.00020646	NB	NB	NB-C
TEST.9	0.254191	0.522688	0.0006668	0.049363	RMS	-	Sk. Muscle
TEST.10	5.77E-07	0.99997 6	5.00E-07	8.37E-05	RMS	RMS	RMS-T
TEST.11	0.626072	0.006254	0.0319496	0.0444521	EWS	-	Prostate Ca.-C
TEST.12	0.999737	5.74E-05	0.00603079	0.00020536	EWS	EWS	EWS-T
TEST.13	0.065958	0.997101	6.42E-05	0.00299387	RMS	-	Sk. Muscle
TEST.14	0.000148	4.80E-07	0.999963	0.00575246	NB	NB	NB-T
TEST.15	7.16E-05	4.53E-06	0.00023765	0.999998	BL	BL	BL-C
TEST.16	0.000144	1.41E-06	0.999993	0.00135978	NB	NB	NB-T
TEST.17	1.15E-05	0.999881	6.79E-05	3.96E-05	RMS	RMS	RMS-T
TEST.18	0.035959	0.003953	0.0105931	0.984807	BL	BL	BL-C
TEST.19	0.999958	6.75E-06	5.64E-05	0.00015527	EWS	EWS	EWS
TEST.20	0.230137	0.016949	4.98E-05	0.0267939	EWS	-	EWS-T
TEST.21	0.99246	0.005905	9.38E-07	0.00121645	EWS	EWS	EWS
TEST.22	1.46E-05	0.999998	2.95E-06	0.00025175	RMS	RMS	RMS-T
TEST.23	0.000559	9.06E-05	0.981119	0.00327278	NB	NB	NB-T
TEST.24	2.20E-05	0.999993	1.05E-05	4.32E-05	RMS	RMS	RMS-T
TEST.25	0.000148	6.42E-08	0.999995	0.00122402	NB	NB	NB-T

Table5.4: The trained network classification and diagnosis results which are compared by real diagnosis results on the right column.

	Presented Study Results			**Khan et al., Paper Results**		
Samples	Classification	Diagnosis	Real Diagnosis	Classification	Diagnosis	Real Diagnosis
TEST.1	NB	NB	NB-C	NB	NB	NB-C
TEST.2	EWS	EWS	EWS-C	EWS	EWS	EWS-C
TEST.3	**EWS**	**-**	**Osteosarcoma-C**	**RMS**	**-**	**Osteosarcoma-C**
TEST.4	RMS	RMS	RMS-T	RMS	RMS	RMS-T
TEST.5	**EWS**	**-**	**Sarcoma**	**NB**	**-**	**Sarcoma**
TEST.6	EWS	EWS	EWS-T	EWS	EWS	EWS-T
TEST.7	BL	BL	BL-C	BL	BL	BL-C
TEST.8	NB	NB	NB-C	NB	NB	NB-C
TEST.9	RMS	-	Sk. Muscle	RMS	-	Sk. Muscle
TEST.10	**RMS**	**RMS**	**RMS-T**	**RMS**	**-**	**RMS-T**
TEST.11	EWS	-	Prostate Ca.-C	EWS	-	Prostate Ca.-C
TEST.12	EWS	EWS	EWS-T	EWS	EWS	EWS-T
TEST.13	RMS	-	Sk. Muscle	RMS	-	Sk. Muscle
TEST.14	NB	NB	NB-T	NB	NB	NB-T
TEST.15	BL	BL	BL-C	BL	BL	BL-C
TEST.16	NB	NB	NB-T	NB	NB	NB-T
TEST.17	RMS	RMS	RMS-T	RMS	RMS	RMS-T
TEST.18	BL	BL	BL-C	BL	BL	BL-C
TEST.19	EWS	EWS	EWS	EWS	EWS	EWS
TEST.20	EWS	-	EWS-T	EWS	-	EWS-T
TEST.21	EWS	EWS	EWS	EWS	EWS	EWS
TEST.22	RMS	RMS	RMS-T	RMS	RMS	RMS-T
TEST.23	NB	NB	NB-T	NB	NB	NB-T
TEST.24	RMS	RMS	RMS-T	RMS	RMS	RMS-T
TEST.25	NB	NB	NB-T	NB	NB	NB-T

Table 5.5: Comparison of the presented study network and Khan et., paper results.

5.4.4 Testing Network with Other Datasets

For the second dataset, Tibshirani et al., (2001) study SRBCT dataset that consist of 43 genes is taken to train and test the presented study network.

For this dataset (88x43) apart from cross validation handling, all procedures mentioned above is implemented identically. While in Khan et al., (2001) has used 3 fold cross validation, Tibshirani et al., (2001) preferred to use 10 fold cross validation and that is why fold number is set as 10 in the presented study as well. The results of this datasets shows that, the Cunernet tool has 100% classification accuracy (including noises samples) (Appendix Table 2) as it has been achieved in Tibshirani et al.,(2001) and 95.4% diagnosis accuracy (Appendix Table 3)

As a third dataset, Pal et al., (2007) SRBCT , with again 88 samples and with only 7 genes, is used on our network and It should be noted that due to the a few number of genes as the Pal et al., study have used multilayer, in the presented study Multilayer Neural Network is also used for this dataset. After training network, the classification result is obtained as 97.7%(Appendix Table 4) over all cancerous samples and also 4 noise samples over 5 is classified correctly. In addition that, all samples are diagnosed by 95.4% accuracy rate (Appendix Table 5).

Even though the accuracy results may differs from studies due to using different methods, the presented study achvieves promising results which proCves that the unnet library is suitable to be used in the field.

5.5 Profiling code

Considering that one of the aim of project is developing a better tool to detect cancer, finding and fixing performance bottlenecks of code is vital .Therefore as it mentioned in *Chapter 4* “Google Performance Analysis Tool” is used in order to find bottleneck of code.

All of the benchmarks and opimisations are done on the workstation NVIDIA K40 GPU (12GB ram and 2880cores) with Ubuntu 14.04, 64 bits 128Gibi RAM and the number of training iteration is set to 100 in each training attempt. Thereafter gperftools CPUProfiler tool is used to gather units/calles numbers of each function/method and according to those result the code is optimised. Thereafter linux *time* command is used to gather average total wall time of software by runnin it several time .

At the first step of gperftool monitoring, 25000 unit is obtained as an average of 5 different run software and in order to improve performance the following changing is implemeted;

- vectors lenght are allocated before storing information
- Function returns are done by move() funtion
- push_back are replaced by emplace_back during value storing to vectors.

Consequentlt unit numbers are reduced to 20000 in average. Afterwards;

- parameters are passed by references to functions rather than passing directly
- Lastly “sigmoid” function is made as inline function.

After these changing, unit number decreased to 19000 and as a result the code optimisation it is made more efficint by around 28%. Furthermore total wall time of software is monitored during these changing and in order to find average total wall time of the software it is run 10 times after each changing. Finally it is obseved that, total spent time is reduced from 105 to 86.5 as each reducing step can be in in Figure 5.10

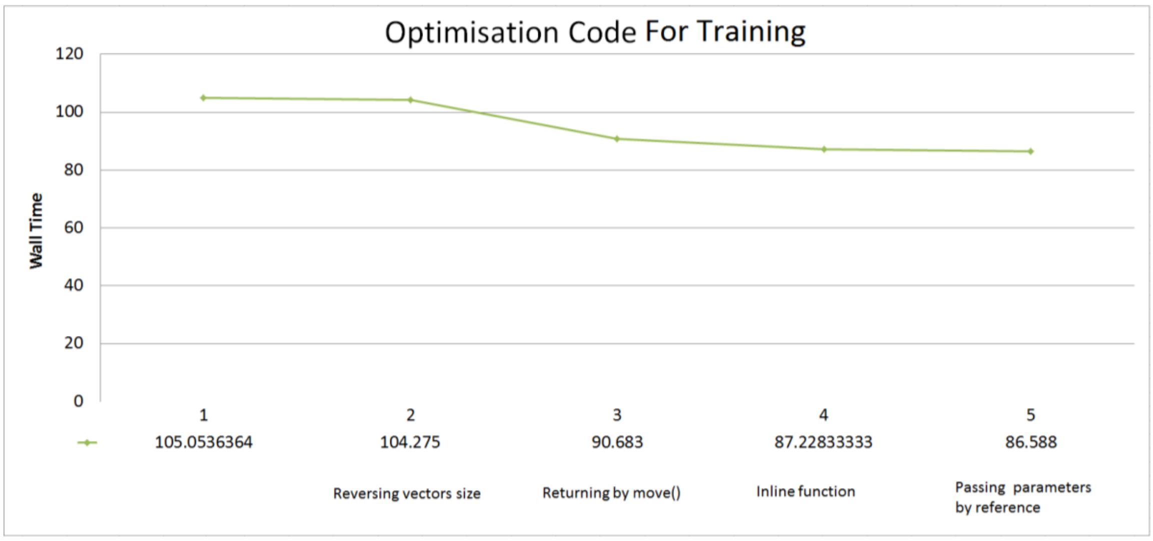


Figure 5.10 : Code optimisation by reversing vector size before storing, returning result by using *move* function , making functions inline and passing parameters by their references

After training optimisation is done, *testing* part of software is optimised by taking classification and diagnosis seperately. During optimisation testing code, same techniques are used with traning optimisations. Such as passing parameter by references,returning values with move function or reserving vectors size before storing. Before and after optimissation techniques are implemeneted, the Cunernet software run 5 times to get average spent wall time of software. At the end of classification optimisation the wall time is reduced from 48.09 to 42.19 and diagnosis wall time is decreased from 132.15 to 123.94.

Furthermore the wall time with different neuron numbers on single layer network is monitored to observe effect of neuron number against to wall time. According to the network input neuron number can be set by passing information on the command panel, the datasets with variety of genes number is created to train network. To create these different dataset, firstly the SRBCT dataset with 2308 genes is downloaded from the web site; http://research.nhgri.nih.gov/microarray/Supplement/ . Thereafter by running the Matlab filtering functions *Genevarfilter*, *Geneentropyfilter*, *Genelowvalfilter*, *and finally by using Generangefilter* repeteadly 2308 genes are reduced gradually and the datasets with 55,127,141,241,368,694 and 1307 genes are stored files in each step to be used.   It should be noted that in regarding to monitoring wall time, SRBCT dataset is not compulsory to be used, any other datasets that have different genes numbers could be used to see effect of neuron number on total time.

After creation different datasets,*train* option is chosen on the Cunernet command line and the dataset name is given one by one with gene number in it. The result of wall times against the used gene numbers shows that increasing neuron number (x axis) is affects the taken time (y axis) significantly on a single layer neural network (Figure 5.11).

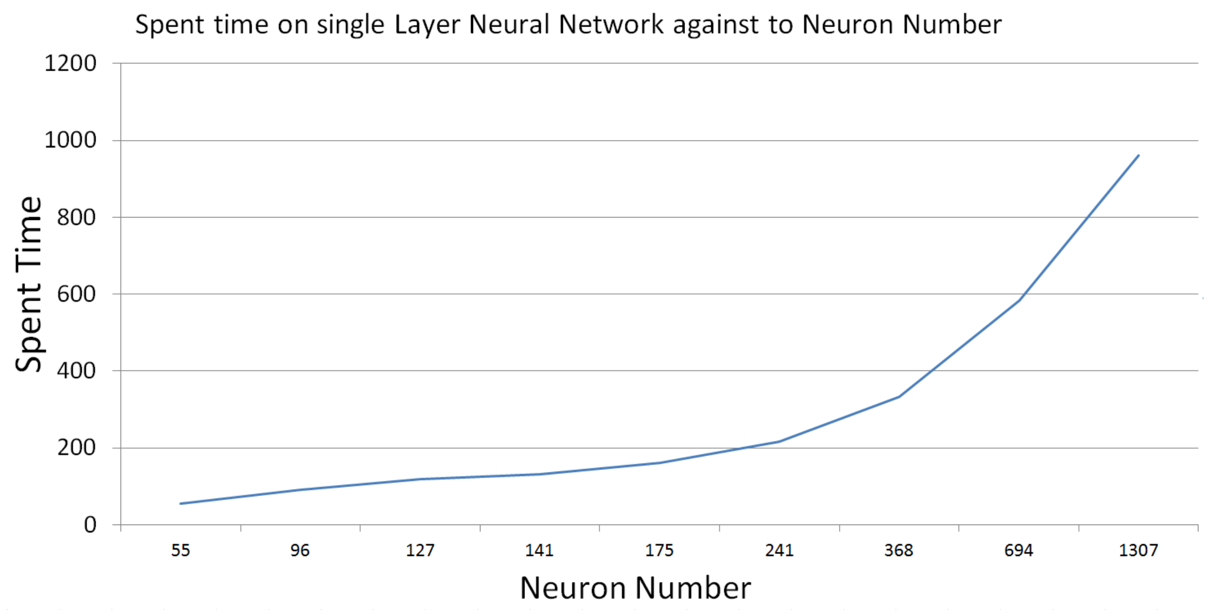


Figure 5.11: The comparison between increasing neuron number and the wall time of running software on single layer neural network.

Thereafter, considering to extensibility of software, the five of the created datasets (mentioned abpve) with 55,93,141,175 and 241 genes is used on Multilayer Neural Network. It is worth the mentioned again that any dataset with different gene numbers, that cause to different neuron numbers, can be used in terms of comparing wall time against to neuron numbers.The created datasets are used with the same training procedures and wall time against to neuron numbers is drawn in Figure 5.12;

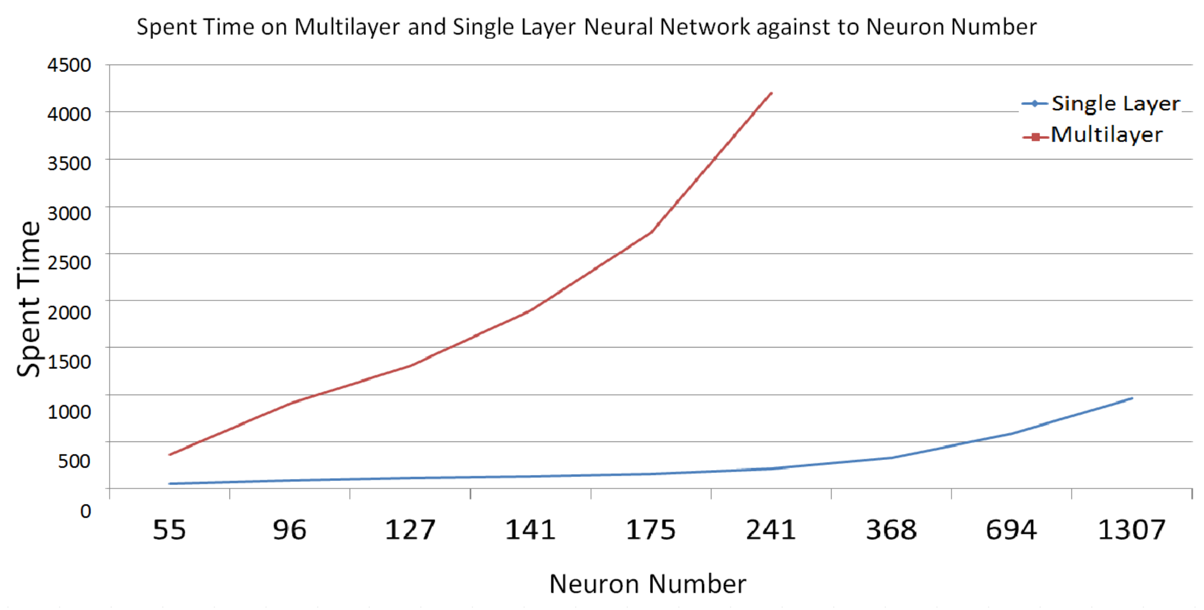


Figure 5.12: The comparison between increasing neuron number against to the spent wall time of the software. The red line represent multilayer neural network implementation and the blue line for single layer neural network.

As it can be seen in Figure 5.12 the neuron number affects spent time enourmously. For the comparison single layer and multi layer network on 241 genes number dataset, it is seen that the time on multilayer neurol network is 20 times more than the time on single layer neurol network.

In addition to neuron number against time, Khan et al., (2001) study dataset with 96 genes on single layer neural network is taken as a sample and the each function time ise examined by using Google Performance Analysis tool to understand which parts can be implemented on CUDA in the future. During using gperftools, the iteration of training process is set to 100 and end of it the the list of functions with spent time on CPU is obtained. According to this list, it is seen that while just inner vector product costs 10% of overall time, the other vector operations (Figure 5.14) cost 47% as it can be seen in Figure 5.13.

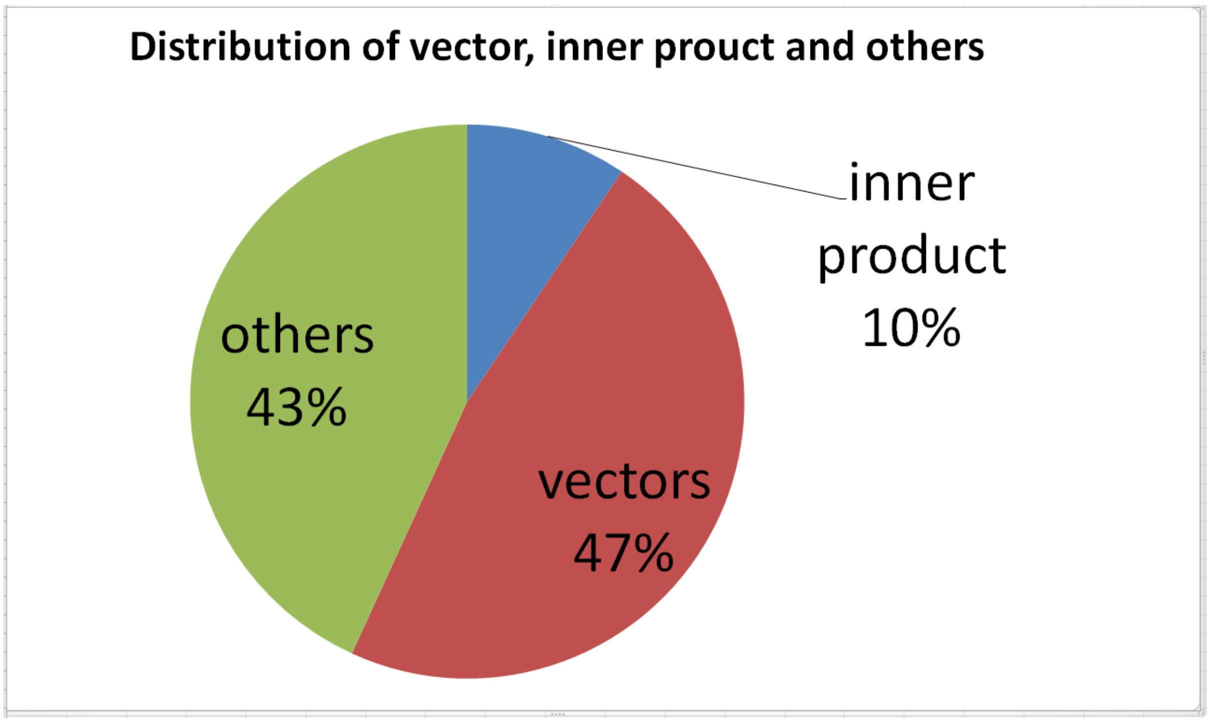


Figure 5.13: The vector operations(red),inner product(blue) proportions in software.

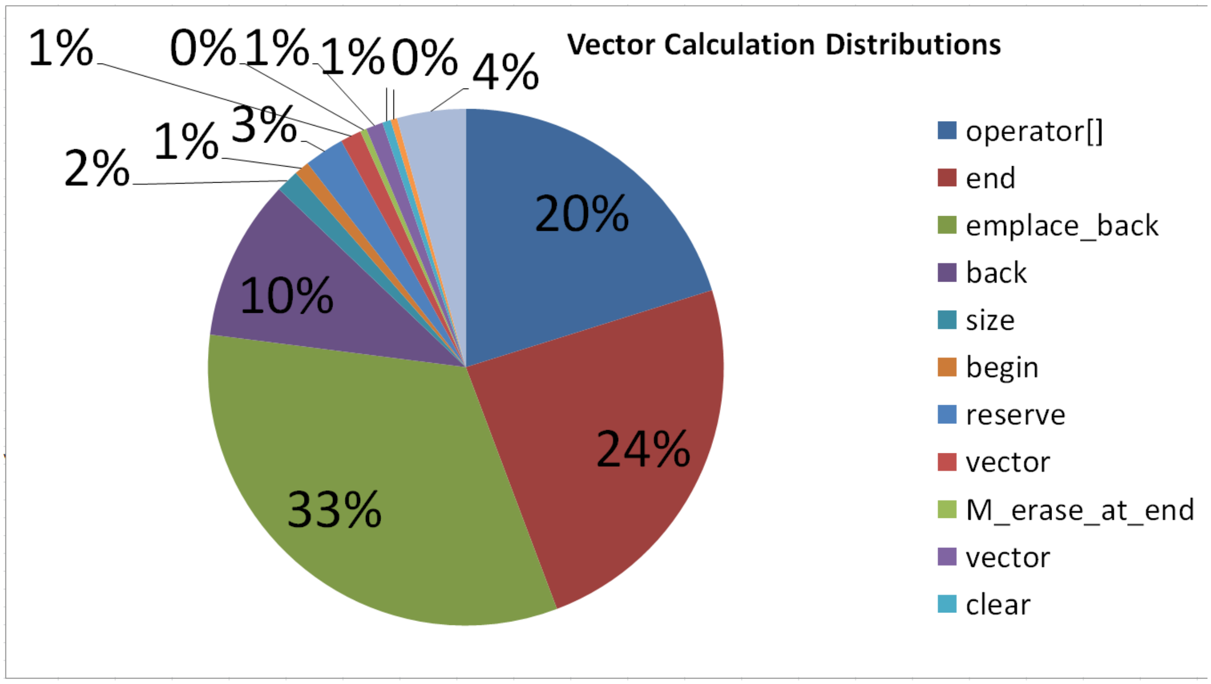


Figure 5.10 Vector operations distrubiton by percentages on all vector calculations.

Regarding to vector operations/calculations,that are generally vector/matrix multiplication with iteration, takes big amount of time in software, these parts can be implemented on CUDA by using Thrust library. It is worth to mention that, the software code is writen by using std vector containers which can be converted to Thrust vector calculation easily. Taking into account that vector calculations/operations takes big amount of time and the code of software is capable of to be converted Thrust, plenty of time can be saved effortlessly by implementing it on CUDA in the future.

Chapter6

Conclusion

This project proposed to develop an open source library called Cunernet which uses gene expression profiles to train artificial neural network system in order to detect cancer information such as metastasis, biomarkers and so on which can help clinicians to tailor cancer treatments effectively.

Differences of this project from other studies can be stated that the used/developed tool will be released as open source and it can be used by any one while it can also be contributed through the Bitbucket project page.

Because the Cunernet artificial neural network library will be able to be used with different datasets, it is designed in a flexible manner in terms of structure (single/ multilayer), learning method (stochastic/batch/semi-batch) and neuron numbers, learning rate and momentum values. In addition, since the project will be implemented on CUDA by using Thrust library in the future, it is written in C++11 in object oriented paradigm with vector containers. Writing code in object oriented paradigm and using vector containers enables code to be implemented on thrust/CUDA easily/effortlessly. Considering that the intention is to release this tool under the BSD3 Licence as an open source, the best development techniques are used along the development. Each code changing step was uploaded to the Bitbucket project page by using git. Also readme file is written to guide user how to compile/run/test the software and instructions are given to show how to contribute to the Cunernet on the project page.

Due to the fact that correct training procedure sequences may vary, a study procedures that has been done by Khan et al., (2001) is mainly followed in the presented study in order to compare result correctness accuracy. Also considering that the dataset may vary in regarding to genes or genes number, SRBCT dataset (88x96) from the same study (Khan et., 2001) is taken to train our neural network and at the end of training/testing stages, our study results are compared with Khan et al., (2001) results.

Even though single layer neural network was trained by SRBCT Khan et al.,(2001) study dataset as it has been done in the Khan et al., (2001) study, multilayer neural network is also used during Pal et al., (2007) study dataset to be trained and tested. Throughout the training network cross validation methods are used to identify and prevent over training and a part of dataset was used to test network correctness known as blind testing.

During the blind testing, Khan et al., (2001) methodology was used; “average vote” calculations for classifications and “distance” calculations for diagnosis. It is worth mentioning that as an extra to Khan et al (2001) procedures, the noise samples were also classified correctly in the classification steps by The Cunernet Library.

After training, the cross validation, classification and diagnosis steps followed those of Khan et al.,(2001), as an extra to Khan’s methods, another blind test (classification and diagnosis) method was implemented to prove the correctness of the network in the present study. The Cunernet library training/testing which was done by using Khan et al.’s dataset (88x96) has shown that the Cunernet library and the Khan et al., study results has same levels of accuracy for both classification and diagnosis. In addition that the network is trained and tested by 2 other SRBCT dataset with 43 and 7 genes from Tibshirani et al., (2001) and Pal et al., (2007) studies respectively. The accuracy f the results obtained was 100% for classification and 95.4% for diagnosis with Tibshirani et al., (2001) study dataset and 97.7% classification and 95.4% diagnosis for Pal e al., (2007) dataset. Even though the non-diagnosed samples may vary due to shuffling dataset randomly, the accuracy percentage of results proves that The Cunernet library is suitable for general use.

After finishing coding,training and testing part in C++11 serial algorithm, the performance monitoring/analysis are done by Google Performance Analysis Tool and linux *time* function. According to result of monitoring the code is optimised with replacing/changing some code parts and also after each code changing the effect of it on the wall time is checked by using *time* function by running software several times. At the end average of wall time is collected and the effectiveness of code optimisation is seen.

6.1 Future works

The Cunernet Library could not be implemented on CUDA by using Thrust library. However by transforming standard vector containers to the Thrust vectors, it can be implemented on CUDA easily and considering from profiling section that, vector calculation takes a part of the running time and implementing vector calculations on CUDA would reduce the spent vector calculations times significantly

Also, in consideration of the fact that end-users will use this tool after releasing, a graphical user interface would make managing library much easier. Furthermore, other artificial neural network features such as online learning algorithms or different activation functions codes can be added and those can be used during training optionally in the future.

Apart from those suggestions, even though the results obtained were satisfactory with 3 different SRBCT datasets, it can still be tested by any other dataset on both single layer or multilayer network to classify different cancer types and also to approve suitability of the Cunernet Library. In addition that since the data format in the project is compatible with Matlab it can be integrated to it easily.

References

Abd El-Rehim, D., Ball, G., Pinder, S., Rakha, E., Paish, C., Robertson, J., Macmillan, D., Blamey, R. and Ellis, I. (2005). High-throughput protein expression analysis using tissue microarray technology of a large well-characterised series identifies biologically distinct classes of breast cancer confirming recent cDNA expression analyses. *International Journal of Cancer*, 116(3), pp.340-350.

Agarwal, D., Kergosien, M., Boocock, D., Rees, R. and Ball, G. (2014). A systems biology approach to identify proliferative biomarkers and pathways in breast cancer. *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp.1-7.

Anfinsen, C. (1973). Principles that Govern the Folding of Protein Chains. *Science*, 181(4096), pp.223-230.

Basu, J. K., Bhattacharyya, D., and Kim, T. H. (2010). Use of artificial neural network in pattern recognition. *International journal of software engineering and its applications*, *4*(2).

Bishop, C. (2006). *Pattern recognition and machine learning*. New York: Springer.

Bray, F., Jemal, A., Grey, N., Ferlay, J. and Forman, D. (2012). Global cancer transitions according to the Human Development Index (2008–2030): a population-based study. *The Lancet Oncology*, 13(8), pp.790-801.

Brown, P. and Botstein, D. (1999). Exploring the new world of the genome with DNA microarrays. *Nat Genet*, 21, pp.33-37.

Chen, Y., Yang, W. and Chiu, H. (2009). Artificial Neural Network Prediction for Cancer Survival Time by Gene Expression Data. *2009 3rd International Conference on Bioinformatics and Biomedical Engineering*, pp.1-4.

Cho, S., and Won, H. (2003). Machine Learning in DNA Microarray Analysis for Cancer Classification. In Y.-P. P. Chen (Ed.), *Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003*, Machine Learning in DNA Microarray Analysis for Cancer Classification, 19, pp.189–198.

Chou, H., Yao, C., Su, S., Lee, C., Hu, K., Terng, H., Shih, Y., Chang, Y., Lu, Y., Chang, C., Wahlqvist, M., Wetter, T. and Chu, C. (2013). Gene expression profiling of breast cancer survivability by pooled cDNA microarray analysis using logistic regression, artificial neural networks and decision trees. *BMC Bioinformatics*, 14(1), p.100.

Chu, C., Yao, C., Chang, Y., Chou, H., Chou, Y., Chen, K., Terng, H., Huang, C., Lee, C., Su, S., Liu, Y., Lin, F., Wetter, T. and Chang, C. (2014). Gene Expression Profiling of Colorectal Tumors and Normal Mucosa by Microarrays Meta-Analysis Using Prediction Analysis of Microarray, Artificial Neural Network, Classification, and Regression Trees. *Disease Markers*, 2014, pp.1-11.

12 Colozza, M., Cardoso, F., Sotiriou, C., Larsimont, D. and Piccart, M. (2005). Bringing Molecular Prognosis and Prediction to the Clinic. *Clinical Breast Cancer*, 6(1), pp.61-76.

13 Cruz, J. A., & Wishart, D. S. (2006). Applications of Machine Learning in Cancer Prediction and Prognosis. *Cancer Informatics*, 2, pp. 59–77.

14 Curtis, C., Shah, S., Chin, S., Turashvili, G., Rueda, O., Dunning, M., Speed, D., Lynch, A., Samarajiwa, S., Yuan, Y., Gräf, S., Ha, G., Haffari, G., Bashashati, A., Russell, R., McKinney, S., Caldas, C., Aparicio, S., Curtis†, C., Shah, S., Caldas, C., Aparicio, S., Brenton, J., Ellis, I., Huntsman, D., Pinder, S., Purushotham, A., Murphy, L., Caldas, C., Aparicio, S., Caldas, C., Bardwell, H., Chin, S., Curtis, C., Ding, Z., Gräf, S., Jones, L., Liu, B., Lynch, A., Papatheodorou, I., Sammut, S., Wishart, G., Aparicio, S., Chia, S., Gelmon, K., Huntsman, D., McKinney, S., Speers, C., Turashvili, G., Watson, P., Ellis, I., Blamey, R., Green, A., Macmillan, D., Rakha, E., Purushotham, A., Gillett, C., Grigoriadis, A., Pinder, S., di Rinaldis, E., Tutt, A., Murphy, L., Parisien, M., Troup, S., Caldas, C., Chin, S., Chan, D., Fielding, C., Maia, A., McGuire, S., Osborne, M., Sayalero, S., Spiteri, I., Hadfield, J., Aparicio, S., Turashvili, G., Bell, L., Chow, K., Gale, N., Huntsman, D., Kovalik, M., Ng, Y., Prentice, L., Caldas, C., Tavaré, S., Curtis, C., Dunning, M., Gräf, S., Lynch, A., Rueda, O., Russell, R., Samarajiwa, S., Speed, D., Markowetz, F., Yuan, Y., Brenton, J., Aparicio, S., Shah, S., Bashashati, A., Ha, G., Haffari, G., McKinney, S., Langerød, A., Green, A., Provenzano, E., Wishart, G., Pinder, S., Watson, P., Markowetz, F., Murphy, L., Ellis, I., Purushotham, A., Børresen-Dale, A., Brenton, J., Tavaré, S., Caldas, C. and Aparicio, S. (2012). The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*.

Dave, S., Wright, G., Tan, B., Rosenwald, A., Gascoyne, R., Chan, W., Fisher, R., Braziel, R., Rimsza, L., Grogan, T., Miller, T., LeBlanc, M., Greiner, T., Weisenburger, D., Lynch, J., Vose, J., Armitage, J., Smeland, E., Kvaloy, S., Holte, H., Delabie, J., Connors, J., Lansdorp, P., Ouyang, Q., Lister, T., Davies, A., Norton, A., Muller-Hermelink, H., Ott, G., Campo, E., Montserrat, E., Wilson, W., Jaffe, E., Simon, R., Yang, L., Powell, J., Zhao, H., Goldschmidt, N., Chiorazzi, M. and Staudt, L. (2004). Prediction of Survival in Follicular Lymphoma Based on Molecular Features of Tumor-Infiltrating Immune Cells. *New England Journal of Medicine*, 351(21), pp.2159-2169.

Floreano, D. and Mattiussi, C. (2008). *Bio-inspired artificial intelligence*. Cambridge, Mass.: MIT Press.

Fortunato, O., Boeri, M., Verri, C., Conte, D., Mensah, M., Suatoni, P., Pastorino, U. and Sozzi, G. (2014). Assessment of Circulating microRNAs in Plasma of Lung Cancer Patients. *Molecules*, 19(3), pp.3038-3054.

Garnett, M., Edelman, E., Heidorn, S., Greenman, C., Dastur, A., Lau, K., Greninger, P., Thompson, I., Luo, X., Soares, J., Liu, Q., Iorio, F., Surdez, D., Chen, L., Milano, R., Bignell, G., Tam, A., Davies, H., Stevenson, J., Barthorpe, S., Lutz, S., Kogera, F., Lawrence, K., McLaren-Douglas, A., Mitropoulos, X., Mironenko, T., Thi, H., Richardson, L., Zhou, W., Jewitt, F., Zhang, T., O'Brien, P., Boisvert, J., Price, S., Hur, W., Yang, W., Deng, X., Butler, A., Choi, H., Chang, J., Baselga, J., Stamenkovic, I., Engelman, J., Sharma, S., Delattre, O., Saez-Rodriguez, J., Gray, N., Settleman, J., Futreal, P., Haber, D., Stratton, M., Ramaswamy, S., McDermott, U. and Benes, C. (2012). Systematic identification of genomic markers of drug sensitivity in cancer cells. *Nature*, 483(7391), pp.570-575.

Geeleher, P., Cox, N. and Huang, R. (2014). Clinical drug response can be predicted using baseline gene expression levels and in vitro drug sensitivity in cell lines. *Genome Biol*, 15(3), p.R47.

Hagan, M., Demuth, H. and Beale, M. (1996). *Neural network design*. Boston: PWS Pub.

Haykin, S. (1994). *Neural networks*. New York: Macmillan.

Heppner, G. H. (1984). Tumor heterogeneity. *Cancer research*, 44(6), pp. 2259-2265.

Incisive Health, (2014). *Saving lives, averting costs*. [online] Incisive Health, pp.Annex 5 page:69. Available at: http://www.incisivehealth.com/uploads/Saving%20lives%20averting%20costs.pdf [Accessed 20 Apr. 2015].

Isa, I., Omar, S., Saad, Z. and Osman, M. (2010). Performance Comparison of Different Multilayer Perceptron Network Activation Functions in Automated Weather Classification. *2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation*, pp.71-75.

Ivshina, A., George, J., Senko, O., Mow, B., Putti, T., Smeds, J., Lindahl, T., Pawitan, Y., Hall, P., Nordgren, H., Wong, J., Liu, E., Bergh, J., Kuznetsov, V. and Miller, L. (2006). Genetic Reclassification of Histologic Grade Delineates New Clinical Subtypes of Breast Cancer. *Cancer Research*, 66(21), pp.10292-10301.

Kan, T., Shimada, Y., Sato, F., Ito, T., Kondo, K., Watanabe, G., Maeda, M., Yamasaki, S., Meltzer, S. and Imamura, M. (2004). Prediction of Lymph Node Metastasis with Use of Artificial Neural Networks Based on Gene Expression Profiles in Esophageal Squamous Cell Carcinoma. *Annals of Surgical Oncology*, 11(12), pp.1070-1078.

Khan, J., Wei, J. S., Ringner, M., Saal, L. H., Ladanyi, M., Westermann, F., ... & Meltzer, P. S. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, *7*(6), 673-679.

Kourou, K., Exarchos, T., Exarchos, K., Karamouzis, M. and Fotiadis, D. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13, pp.8-17.

Lancashire, L., Powe, D., Reis-Filho, J., Rakha, E., Lemetre, C., Weigelt, B., Abdel-Fatah, T., Green, A., Mukta, R., Blamey, R., Paish, E., Rees, R., Ellis, I. and Ball, G. (2009). A validated gene expression profile for detecting clinical outcome in breast cancer using artificial neural networks. *Breast Cancer Res Treat*, 120(1), pp.83-93.

Maltarollo, Vinícius Gonçalves, Albérico Borges Ferreira da Silva, and Káthia Maria Honório. *Applications of artificial neural networks in chemical problems*. INTECH Open Access Publisher, 2013.

MATLAB and Bioinformatics Toolbox Release R2014a, The MathWorks, Inc., Natick, Massachusetts, United States.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4), pp.115-133.

McLendon, R., Friedman, A., Bigner, D., Van Meir, E., Brat, D., M. Mastrogianakis, G., Olson, J., Mikkelsen, T., Lehman, N., Aldape, K., Alfred Yung, W., Bogler, O., VandenBerg, S., Berger, M., Prados, M., Muzny, D., Morgan, M., Scherer, S., Sabo, A., Nazareth, L., Lewis, L., Hall, O., Zhu, Y., Ren, Y., Alvi, O., Yao, J., Hawes, A., Jhangiani, S., Fowler, G., San Lucas, A., Kovar, C., Cree, A., Dinh, H., Santibanez, J., Joshi, V., Gonzalez-Garay, M., Miller, C., Milosavljevic, A., Donehower, L., Wheeler, D., Gibbs, R., Cibulskis, K., Sougnez, C., Fennell, T., Mahan, S., Wilkinson, J., Ziaugra, L., Onofrio, R., Bloom, T., Nicol, R., Ardlie, K., Baldwin, J., Gabriel, S., Lander, E., Ding, L., Fulton, R., McLellan, M., Wallis, J., Larson, D., Shi, X., Abbott, R., Fulton, L., Chen, K., Koboldt, D., Wendl, M., Meyer, R., Tang, Y., Lin, L., Osborne, J., Dunford-Shore, B., Miner, T., Delehaunty, K., Markovic, C., Swift, G., Courtney, W., Pohl, C., Abbott, S., Hawkins, A., Leong, S., Haipek, C., Schmidt, H., Wiechert, M., Vickery, T., Scott, S., Dooling, D., Chinwalla, A., Weinstock, G., Mardis, E., Wilson, R., Getz, G., Winckler, W., Verhaak, R., Lawrence, M., O'Kelly, M., Robinson, J., Alexe, G., Beroukhim, R., Carter, S., Chiang, D., Gould, J., Gupta, S., Korn, J., Mermel, C., Mesirov, J., Monti, S., Nguyen, H., Parkin, M., Reich, M., Stransky, N., Weir, B., Garraway, L., Golub, T., Meyerson, M., Chin, L., Protopopov, A., Zhang, J., Perna, I., Aronson, S., Sathiamoorthy, N., Ren, G., Yao, J., Wiedemeyer, W., Kim, H., Won Kong, S., Xiao, Y., Kohane, I., Seidman, J., Park, P., Kucherlapati, R., Laird, P., Cope, L., Herman, J., Weisenberger, D., Pan, F., Van Den Berg, D., Van Neste, L., Mi Yi, J., Schuebel, K., Baylin, S., Absher, D., Li, J., Southwick, A., Brady, S., Aggarwal, A., Chung, T., Sherlock, G., Brooks, J., Myers, R., Spellman, P., Purdom, E., Jakkula, L., Lapuk, A., Marr, H., Dorton, S., Gi Choi, Y., Han, J., Ray, A., Wang, V., Durinck, S., Robinson, M., Wang, N., Vranizan, K., Peng, V., Van Name, E., Fontenay, G., Ngai, J., Conboy, J., Parvin, B., Feiler, H., Speed, T., Gray, J., Brennan, C., Socci, N., Olshen, A., Taylor, B., Lash, A., Schultz, N., Reva, B., Antipin, Y., Stukalov, A., Gross, B., Cerami, E., Qing Wang, W., Qin, L., Seshan, V., Villafania, L., Cavatore, M., Borsu, L., Viale, A., Gerald, W., Sander, C., Ladanyi, M., Perou, C., Neil Hayes, D., Topal, M., Hoadley, K., Qi, Y., Balu, S., Shi, Y., Wu, J., Penny, R., Bittner, M., Shelton, T., Lenkiewicz, E., Morris, S., Beasley, D., Sanders, S., Kahn, A., Sfeir, R., Chen, J., Nassau, D., Feng, L., Hickey, E., Zhang, J., Weinstein, J., Barker, A., Gerhard, D., Vockley, J., Compton, C., Vaught, J., Fielding, P., Ferguson, M., Schaefer, C., Madhavan, S., Buetow, K., Collins, F., Good, P., Guyer, M., Ozenberger, B., Peterson, J. and Thomson, E. (2008). Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, 455(7216), pp.1061-1068.

Minsky, M. and Papert, S. (1969). *Perceptrons; an introduction to computational geometry*. Cambridge, Mass.: MIT Press.

National Cancer Institute,. (2015). *What Is Cancer?*. Retrieved 15 April 2015, from http://www.cancer.gov/cancertopics/what-is-cancer

Pal, N., Aguan, K., Sharma, A. and Amari, S. (2007). Discovering biomarkers from gene expression data for predicting cancer subgroups using neural networks and relational fuzzy clustering. *BMC Bioinformatics*, 8(1), p.5.

Prat, A., Bianchini, G., Thomas, M., Belousov, A., Cheang, M., Koehler, A., Gomez, P., Semiglazov, V., Eiermann, W., Tjulandin, S., Byakhow, M., Bermejo, B., Zambetti, M., Vazquez, F., Gianni, L. and Baselga, J. (2014). Research-Based PAM50 Subtype Predictor Identifies Higher Responses and Improved Survival Outcomes in HER2-Positive Breast Cancer in the NOAH Study. *Clinical Cancer Research*, 20(2), pp.511-521.

Ross, D., Scherf, U., Eisen, M., Perou, C., Rees, C., Spellman, P., Iyer, V., Jeffrey, S., Van de Rijn, M., Waltham, M., Pergamenschikov, A., Lee, J., Lashkari, D., Shalon, D., Myers, T., Weinstein, J., Botstein, D. and Brown, P. (2000). Systematic variation in gene expression patterns in human cancer cell lines. *Nat Genet*, 24(3), pp.227-235.

Rumelhart, D. and McClelland, J. (1986). *Parallel distributed processing*. Cambridge, Mass.: MIT Press.

Russell, S. and Norvig, P. (1995). *Artificial intelligence*. Englewood Cliffs, N.J.: Prentice Hall.

Schena, M., Shalon, D., Davis, R. and Brown, P. (1995). Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray. *Science*, 270(5235), pp.467-470.

Shipp, M., Ross, K., Tamayo, P., Weng, A., Kutok, J., Aguiar, R., Gaasenbeek, M., Angelo, M., Reich, M., Pinkus, G., Ray, T., Koval, M., Last, K., Norton, A., Lister, T., Mesirov, J., Neuberg, D., Lander, E., Aster, J. and Golub, T. (2002). Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature Medicine*, 8(1), pp.68-74.

Sibi, P., Jones, S. A., & Siddarth, P. (2013). Analysis of Different Activation Functions Using Back Propagation Neural Networks. *Journal of Theoretical and Applied Information Technology*, *47*(3), pp.1264-1268.

Stewart BW, Wild CP, 2014. *World Cancer Report 2014*. Lyon, France: International Agency for Research on Cancer.

Sotiriou, C. and Piccart, M. (2007). Taking gene-expression profiling to the clinic: when will molecular signatures become relevant to patient care?. *Nat Rev Cancer*, 7(7), pp.545-553.

Tibshirani, R., Hastie, T., Narasimhan, B. and Chu, G. (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10), pp.6567-6572.

Wei, S., Paik, H., Kim, Y., Baldwin, R., Liyanarachchi, S., Li, L., Wang, Z., Wan, J., Davuluri, R., Karlan, B., Gifford, G., Brown, R., Kim, S., Huang, T. and Nephew, K. (2006). Prognostic DNA Methylation Biomarkers in Ovarian Cancer. *Clinical Cancer Research*, 12(9), pp.2788-2794.

WIDROW, B. and HOFF, M. (1960). *ADAPTIVE SWITCHING CIRCUITS*. Ft. Belvoir: Defense Technical Information Center.

www1, (2015). *Worldwide cancer mortality statistics : Cancer Research UK*. [online]Cancerresearchuk.org. Available at: http://www.cancerresearchuk.org/cancer-info/cancerstats/world/mortality/ [Accessed 9 May 2015].

www2, (2015). *Fact Sheets by Cancer*. [online] Globocan.iarc.fr. Available at: http://globocan.iarc.fr/Pages/fact_sheets_cancer.aspx [Accessed 10 May 2015].

www3, (2015). *What are the different types of cancer treatment?*. [online] Cancer.org. Available at: http://www.cancer.org/treatment/understandingyourdiagnosis/talkingaboutcancer/whensomeoneyouworkwithhascancer/when-someone-you-work-with-has-cancer-questions-and-answers-about-cancer-treatment [Accessed 10 May 2015].

www4, (2015). *WHO | Cancer*. [online] Who.int. Available at: http://www.who.int/mediacentre/factsheets/fs297/en/ [Accessed 9 May 2015].

www5, (2015). *WHO | Early detection of cancer*. [online] Who.int. Available at: http://www.who.int/cancer/detection/en/ [Accessed 9 May 2015].

www6, (2014). *Stages of cancer*. [online] Cancer Research UK. Available at: http://www.cancerresearchuk.org/about-cancer/what-is-cancer/stages-of-cancer [Accessed 9 May 2015].

www7, (2015). [online] Available at: http://www.cancerresearchuk.org/sites/default/files/cruk_research_strategy.pdf [Accessed 9 May 2015].

www8, (2015). *DNA Is a Structure That Encodes Biological Information | Learn Science at Scitable*. [online] Nature.com. Available at: http://www.nature.com/scitable/topicpage/DNA-Is-a-Structure-that-Encodes-Information-6493050 [Accessed 9 May 2015].

www9, (2015). *Translation: DNA to mRNA to Protein | Learn Science at Scitable*. [online] Nature.com. Available at: http://www.nature.com/scitable/topicpage/translation-dna-to-mrna-to-protein-393 [Accessed 9 May 2015].

www10, (2014). *Gene expression profiling : Latest content : nature.com*. [online] Nature.com. Available at: http://www.nature.com/subjects/gene-expression-profiling [Accessed 9 May 2015].

www11, (2015). *Leukemia - Figure 1 for article: DNA microarrays for comparison of gene expression profiles between diagnosis and relapse in precursor-B acute lymphoblastic leukemia: choice of technique and purification influence the identification of potential diagnostic markers*. [online] Nature.com. Available at: http://www.nature.com/leu/journal/v17/n7/fig_tab/2402974f1.html [Accessed 9 May 2015].

WWW12, (2015). *Talking Glossary of Genetic Terms*. [online] Genome.gov. Available at: http://www.genome.gov/glossary/ [Accessed 10 May 2015].

WWW13 (2014). *How cancer starts*. [online] Available at: http://www.cancerresearchuk.org/about-cancer/what-is-cancer/how-cancer-starts [Accessed 15 Aug. 2015].

www14 Rise.duke.edu, (2015). [online] Available at: http://rise.duke.edu/seek/pages/page.html?0205 [Accessed 15 Aug. 2015].

www15 Cancer Research UK, (2015). *Cancer incidence for common cancers*. [online] Available at: http://www.cancerresearchuk.org/health-professional/cancer-statistics/incidence/common-cancers-compared#heading-Zero [Accessed 16 Aug. 2015].

www 16Ncbi.nlm.nih.gov, (2015). *Home - GEO - NCBI*. [online] Available at: http://www.ncbi.nlm.nih.gov/geo/ [Accessed 18 Aug. 2015].

www17 Tcga-data.nci.nih.gov, (2015). *The Cancer Genome Atlas - Data Portal*. [online] Available at: https://tcga-dta.nci.nih.gov/tcga/ [Accessed 18 Aug. 2015].

www18, Webpages.ttu.edu, (2015). *Neural Network Basics*. [online] Available at: http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html [Accessed 26 Aug. 2015].

Xu, J., Reumers, J., Couceiro, J., De Smet, F., Gallardo, R., Rudyak, S., Cornelis, A., Rozenski, J., Zwolinska, A., Marine, J., Lambrechts, D., Suh, Y., Rousseau, F. and Schymkowitz, J. (2011). Gain of function of mutant p53 by coaggregation with multiple tumor suppressors. *Nature Chemical Biology*, 7(5), pp.285-295.

Xu, Q., Liang, Y. and Du, Y. (2004). Monte Carlo cross-validation for selecting a model and estimating the prediction error in multivariate calibration. *Journal of Chemometrics*, 18(2), pp.112-120.

Yang, Y. (2002). Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research*, 30(4), pp.15e-15.

Yu, L., & Liu, H. (2003, August). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML* (Vol. 3, pp. 856-863).

Appendices

SAMPLES		Distance	Network Diagnosis	Real Diagnosis	Diagnosis Result
Sample	1	4.77E-06	EWS	EWS	MATCHED
Sample	2	9.16E-06	EWS	EWS	MATCHED
Sample	3	3.35E-06	EWS	EWS	MATCHED
Sample	4	1.22E-05	EWS	EWS	MATCHED
Sample	5	5.71E-08	EWS	EWS	MATCHED
Sample	6	1.48E-07	EWS	EWS	MATCHED
Sample	7	7.23E-08	EWS	EWS	MATCHED
Sample	8	8.87E-09	EWS	EWS	MATCHED
Sample	9	6.50E-09	EWS	EWS	MATCHED
Sample	10	0.00103251	EWS	EWS	MATCHED
Sample	11	4.21E-06	EWS	EWS	MATCHED
Sample	12	9.06E-06	EWS	EWS	MATCHED
Sample	13	3.37E-07	EWS	EWS	MATCHED
Sample	14	1.36E-07	EWS	EWS	MATCHED
Sample	15	0.000315283	EWS	EWS	MATCHED
Sample	16	0.0176963	EWS	EWS	MATCHED
Sample	17	0.0012619	EWS	EWS	MATCHED
Sample	18	0.00033273	EWS	EWS	MATCHED
Sample	19	7.66E-05	EWS	EWS	MATCHED
Sample	20	1.76E-06	EWS	EWS	MATCHED
Sample	21	0.0135463	EWS	EWS	MATCHED
Sample	22	0.00329583	EWS	EWS	MATCHED
Sample	23	0.00499266	EWS	EWS	MATCHED
Sample	24	0.00107803	BL	BL	MATCHED
Sample	25	0.00293074	BL	BL	MATCHED
Sample	26	0.000392999	BL	BL	MATCHED
Sample	27	0.000366348	BL	BL	MATCHED
Sample	28	0.00272902	BL	BL	MATCHED
Sample	29	0.00463481	BL	BL	MATCHED
Sample	30	0.000699677	BL	BL	MATCHED
Sample	31	0.00312532	BL	BL	MATCHED
Sample	32	4.00019	NB	NB	MATCHED
Sample	33	4.09755	NB	NB	MATCHED
Sample	34	5.94444	NB	NB	MATCHED
Sample	35	3.99983	NB	NB	MATCHED
Sample	36	4.00028	NB	NB	MATCHED
Sample	37	4.46107	NB	NB	MATCHED
Sample	38	4.00012	NB	NB	MATCHED
Sample	39	3.99991	NB	NB	MATCHED
Sample	40	4.00185	NB	NB	MATCHED
Sample	41	4.01289	NB	NB	MATCHED
Sample	42	4.00298	NB	NB	MATCHED
Sample	43	4.00049	NB	NB	MATCHED
Sample	44	1.35E-05	RMS	RMS	MATCHED
Sample	45	0.000232219	RMS	RMS	MATCHED
Sample	46	4.15E-06	RMS	RMS	MATCHED
Sample	47	0.000196219	RMS	RMS	MATCHED
Sample	48	0.00818687	RMS	RMS	MATCHED
Sample	49	0.0150234	RMS	RMS	MATCHED
Sample	50	0.0019113	RMS	RMS	MATCHED
Sample	51	0.00366908	RMS	RMS	MATCHED
Sample	52	0.00145155	RMS	RMS	MATCHED
Sample	53	0.00322221	RMS	RMS	MATCHED
Sample	54	6.94E-06	RMS	RMS	MATCHED
Sample	55	0.000213505	RMS	RMS	MATCHED
Sample	56	0.00076945	RMS	RMS	MATCHED
Sample	57	4.29E-06	RMS	RMS	MATCHED
Sample	58	0.0053966	RMS	RMS	MATCHED
Sample	59	0.000808874	RMS	RMS	MATCHED
Sample	60	2.94E-08	RMS	RMS	MATCHED
Sample	61	0.0137728	RMS	RMS	MATCHED
Sample	62	0.0282319	RMS	RMS	MATCHED
Sample	63	1.38E-14	RMS	RMS	MATCHED

TESTS		Distance	Network Diagnosis	Real Diagnosis	Diagnosis Result
(Test	1)	12.3335	NB	NB	MATCHED
(Test	2)	0.00134764	EWS	EWS	MATCHED
(Test	3)	570.176	NOISE	NOISE	MATCHED
(Test	4)	6.03E-08	RMS	RMS	MATCHED
(Test	5)	25.2863	NOISE	NOISE	MATCHED
(Test	6)	0.000567549	EWS	EWS	MATCHED
(Test	7)	0.00882391	BL	BL	MATCHED
(Test	8)	11.9997	NB	NB	MATCHED
(Test	9)	814.101	NOISE	NOISE	MATCHED
(Test	10)	5.32E-06	RMS	RMS	MATCHED
(Test	11)	392.901	NOISE	NOISE	MATCHED
(Test	12)	0.186486	NOISE	EWS	No Matched
(Test	13)	2784.12	NOISE	NOISE	MATCHED
(Test	14)	12.0148	NB	NB	MATCHED
(Test	15)	0.00393535	BL	BL	MATCHED
(Test	16)	12	NB	NB	MATCHED
(Test	17)	1.77E-05	RMS	RMS	MATCHED
(Test	18)	0.0578823	NOISE	BL	No Matched
(Test	19)	0.000500264	EWS	EWS	MATCHED
(Test	20)	43.1108	NOISE	EWS	No Matched
(Test	21)	0.0224923	EWS	EWS	MATCHED
(Test	22)	4.19E-06	RMS	RMS	MATCHED
(Test	23)	12.512	NB	NB	MATCHED
(Test	24)	2.93E-05	RMS	RMS	MATCHED
(Test	25)	12.0002	NB	NB	MATCHED

Table 1: Diagnostic Classification :by using 96 genes dataset from Khan et al., (2001) study.

		Real Output	Network Output	Classification	Network Outputs			
Sample	1	0	0	Matched	1	0.000119	1.22E-06	5.06E-05
Sample	2	0	0	Matched	0.999993	1.02E-05	1.75E-05	0.000115
Sample	3	0	0	Matched	0.999991	0.000158	1.52E-05	3.96E-05
Sample	4	0	0	Matched	0.999984	0.000408	2.02E-07	0.000367
Sample	5	0	0	Matched	0.999997	9.15E-05	2.28E-06	9.32E-05
Sample	6	0	0	Matched	1	3.43E-06	2.06E-06	9.58E-05
Sample	7	0	0	Matched	1	6.84E-06	9.96E-08	4.42E-06
Sample	8	0	0	Matched	1	2.37E-06	7.58E-07	5.91E-05
Sample	9	0	0	Matched	1	8.47E-07	5.21E-07	4.35E-07
Sample	10	0	0	Matched	0.998486	0.000787	3.13E-07	9.66E-05
Sample	11	0	0	Matched	0.999999	8.58E-06	3.70E-05	9.61E-05
Sample	12	0	0	Matched	1	5.40E-06	1.54E-05	3.39E-06
Sample	13	0	0	Matched	1	3.88E-06	4.40E-06	9.49E-06
Sample	14	0	0	Matched	0.999994	6.20E-06	3.43E-05	1.50E-05
Sample	15	0	0	Matched	0.999893	2.62E-05	8.45E-05	0.000176
Sample	16	0	0	Matched	0.99862	0.000638	2.67E-05	0.000875
Sample	17	0	0	Matched	0.998552	0.000427	0.001277	0.000481
Sample	18	0	0	Matched	0.999936	0.000165	0.000231	0.000366
Sample	19	0	0	Matched	0.999987	9.83E-05	9.01E-05	0.000136
Sample	20	0	0	Matched	1	2.45E-05	1.37E-05	8.76E-07
Sample	21	0	0	Matched	0.999832	5.62E-06	0.000629	7.36E-05
Sample	22	0	0	Matched	0.999751	0.000246	0.000734	0.000308
Sample	23	0	0	Matched	0.997778	0.000517	0.001197	0.000594
Sample	24	3	3	Matched	2.66E-05	2.11E-05	3.14E-05	0.999983
Sample	25	3	3	Matched	9.97E-05	2.66E-05	0.000187	0.999893
Sample	26	3	3	Matched	0.000206728	0.000337	0.000343	0.999801
Sample	27	3	3	Matched	0.00263174	0.001841	0.001977	0.997815

Sample	28	3	3	Matched	0.000168041	0.000559	4.29E-05	0.999929
Sample	29	3	3	Matched	0.00238886	0.002353	0.001782	0.997295
Sample	30	3	3	Matched	0.000133111	0.000291	2.48E-05	0.999913
Sample	31	3	3	Matched	0.000613028	0.000737	0.000641	0.998694
Sample	32	2	2	Matched	1.03E-05	0.000152	0.999709	2.11E-06
Sample	33	2	2	Matched	2.46E-05	0.000814	0.999554	8.77E-05
Sample	34	2	2	Matched	2.65E-07	0.001683	0.999103	8.02E-08
Sample	35	2	2	Matched	0.000556433	0.000324	0.99904	0.001313
Sample	36	2	2	Matched	2.71E-05	2.04E-05	0.999988	0.000114
Sample	37	2	2	Matched	0.00146565	0.00116	0.997142	0.00087
Sample	38	2	2	Matched	9.35E-05	4.91E-05	0.999216	0.000642
Sample	39	2	2	Matched	4.84E-06	9.91E-06	0.999991	9.61E-05
Sample	40	2	2	Matched	3.50E-05	0.000471	0.999519	0.000113
Sample	41	2	2	Matched	0.00238453	0.000399	0.99856	0.000241
Sample	42	2	2	Matched	0.000636234	5.00E-05	0.999616	0.000243
Sample	43	2	2	Matched	5.44E-06	0.001867	0.999313	2.26E-05
Sample	44	1	1	Matched	1.41E-07	0.999998	1.92E-06	5.50E-08
Sample	45	1	1	Matched	0.000214327	0.99985	2.98E-05	1.18E-05
Sample	46	1	1	Matched	3.61E-05	0.999969	3.80E-05	1.30E-06
Sample	47	1	1	Matched	2.07E-05	0.997515	0.001829	7.53E-07
Sample	48	1	1	Matched	0.000134179	0.999845	5.04E-05	4.79E-05
Sample	49	1	1	Matched	1.11E-05	0.999234	0.000953	0.000185
Sample	50	1	1	Matched	3.58E-05	0.999525	0.000636	3.14E-05
Sample	51	1	1	Matched	0.000453	0.998465	0.000143	0.000494
Sample	52	1	1	Matched	1.24E-05	0.999647	0.000825	7.90E-06
Sample	53	1	1	Matched	0.000153873	0.998548	9.64E-05	0.000235
Sample	54	1	1	Matched	8.08E-07	0.999996	7.17E-08	9.88E-05
Sample	55	1	1	Matched	1.01E-05	0.999974	4.55E-08	2.73E-06
Sample	56	1	1	Matched	0.000572478	0.999948	6.06E-07	0.000847
Sample	57	1	1	Matched	5.25E-06	0.999986	2.86E-06	1.05E-08
Sample	58	1	1	Matched	1.21E-06	0.999998	3.18E-08	4.53E-06
Sample	59	1	1	Matched	3.42E-05	0.999939	1.96E-06	3.48E-08
Sample	60	1	1	Matched	2.48E-06	0.999999	2.62E-07	1.47E-07
Sample	61	1	1	Matched	0.00021953	0.99987	7.63E-07	2.28E-05
Sample	62	1	1	Matched	0.000672211	0.9994	6.58E-06	4.04E-07
Sample	63	1	1	Matched	2.57E-08	0.999989	5.04E-07	1.53E-15
Sample	64	2	2	Matched	0.0154646	7.75E-05	0.992284	0.000267
Sample	65	0	0	Matched	0.999794	0.000487	0.000186	1.20E-05
Sample	66	-1	-1	Matched	0.598503	0.001732	0.053345	0.022861
Sample	67	1	1	Matched	8.33E-08	1	1.09E-07	4.17E-09
Sample	68	-1	-1	Matched	0.116673	0.016672	0.005149	0.137668
Sample	69	0	0	Matched	1	1.22E-06	4.78E-05	9.60E-06
Sample	70	3	3	Matched	0.000403787	7.40E-05	8.62E-06	0.999981
Sample	71	2	2	Matched	4.04E-05	3.75E-05	0.999868	8.61E-05
Sample	72	-1	-1	Matched	0.0469618	0.841103	3.47E-07	0.865525
Sample	73	1	1	Matched	8.80E-08	0.999983	3.39E-08	4.86E-09
Sample	74	-1	-1	Matched	0.413516	0.026146	0.003504	0.184362
Sample	75	0	0	Matched	0.99998	6.58E-06	2.58E-06	0.005912
Sample	76	-1	-1	Matched	0.00659524	0.984814	9.44E-11	0.992397
Sample	77	2	2	Matched	2.44E-06	4.59E-06	0.99975	0.005545
Sample	78	3	3	Matched	9.52E-05	5.44E-05	0.000135	0.999953
Sample	79	2	2	Matched	3.78E-07	4.17E-07	0.999999	0.000112
Sample	80	1	1	Matched	4.76E-05	0.999933	1.12E-06	9.17E-06
Sample	81	3	3	Matched	0.022206	0.003932	0.000773	0.99497
Sample	82	0	0	Matched	1	0.002072	1.29E-07	0.000119
Sample	83	0	0	Matched	0.278143	0.042726	0.000137	0.001357
Sample	84	0	0	Matched	0.998408	0.000829	3.45E-07	0.000104
Sample	85	1	1	Matched	1.58E-06	0.999999	1.35E-08	0.000709
Sample	86	2	2	Matched	2.25E-07	2.01E-06	0.999744	7.12E-05
Sample	87	1	1	Matched	7.11E-07	0.999994	1.05E-06	2.68E-07
Sample	88	2	2	Matched	7.78E-07	2.37E-07	0.999997	0.000317

Table 2: Classification Result , the SRBCT dataset with 43 genes from Tibshirani et al., (2001) study is classified with 100% accuracy by using 10 fold cross validation

		Distance	Network Diagnosis	Real Diagnosis	Diagnosis Result
Sample	1	1.45E-06	EWS	EWS	MATCHED
Sample	2	5.06E-06	EWS	EWS	MATCHED
Sample	3	2.24E-05	EWS	EWS	MATCHED
Sample	4	0.000119311	EWS	EWS	MATCHED
Sample	5	3.60E-06	EWS	EWS	MATCHED
Sample	6	8.40E-07	EWS	EWS	MATCHED
Sample	7	1.01E-07	EWS	EWS	MATCHED
Sample	8	6.04E-07	EWS	EWS	MATCHED
Sample	9	9.57E-10	EWS	EWS	MATCHED
Sample	10	0.0302619	EWS	EWS	MATCHED
Sample	11	1.49E-06	EWS	EWS	MATCHED
Sample	12	4.41E-08	EWS	EWS	MATCHED
Sample	13	4.20E-07	EWS	EWS	MATCHED
Sample	14	1.84E-05	EWS	EWS	MATCHED
Sample	15	0.00018317	EWS	EWS	MATCHED
Sample	16	0.00617019	EWS	EWS	MATCHED
Sample	17	0.00164225	EWS	EWS	MATCHED
Sample	18	0.000128348	EWS	EWS	MATCHED
Sample	19	3.25E-05	EWS	EWS	MATCHED
Sample	20	2.52E-06	EWS	EWS	MATCHED
Sample	21	0.000173172	EWS	EWS	MATCHED
Sample	22	0.00126992	EWS	EWS	MATCHED
Sample	23	0.00581252	EWS	EWS	MATCHED
Sample	24	0.000173586	BL	BL	MATCHED
Sample	25	0.000588761	BL	BL	MATCHED
Sample	26	0.00241238	BL	BL	MATCHED
Sample	27	0.0149355	BL	BL	MATCHED
Sample	28	0.000271029	BL	BL	MATCHED
Sample	29	0.0238363	BL	BL	MATCHED
Sample	30	0.000402575	BL	BL	MATCHED
Sample	31	0.00228334	BL	BL	MATCHED
Sample	32	0.00219025	NB	NB	MATCHED
Sample	33	0.0336043	NB	NB	MATCHED
Sample	34	0.265397	NB	NB	MATCHED
Sample	35	0.00556229	NB	NB	MATCHED
Sample	36	0.000117115	NB	NB	MATCHED
Sample	37	0.0692739	NB	NB	MATCHED
Sample	38	0.00562344	NB	NB	MATCHED
Sample	39	0.000107489	NB	NB	MATCHED
Sample	40	0.00214072	NB	NB	MATCHED
Sample	41	0.00553467	NB	NB	MATCHED
Sample	42	0.000731026	NB	NB	MATCHED
Sample	43	0.0156168	NB	NB	MATCHED
Sample	44	1.56E-07	RMS	RMS	MATCHED
Sample	45	0.000445586	RMS	RMS	MATCHED
Sample	46	1.19E-05	RMS	RMS	MATCHED
Sample	47	0.00384546	RMS	RMS	MATCHED
Sample	48	8.81E-05	RMS	RMS	MATCHED
Sample	49	0.00166123	RMS	RMS	MATCHED
Sample	50	0.00048605	RMS	RMS	MATCHED
Sample	51	0.0349973	RMS	RMS	MATCHED
Sample	52	0.000384416	RMS	RMS	MATCHED
Sample	53	0.0753344	RMS	RMS	MATCHED
Sample	54	3.91E-06	RMS	RMS	MATCHED

Sample 55	1.56E-05	RMS	RMS	MATCHED
Sample 56	0.000248781	RMS	RMS	MATCHED
Sample 57	5.19E-07	RMS	RMS	MATCHED
Sample 58	1.92E-07	RMS	RMS	MATCHED
Sample 59	3.07E-06	RMS	RMS	MATCHED
Sample 60	1.17E-07	RMS	RMS	MATCHED
Sample 61	3.14E-05	RMS	RMS	MATCHED
Sample 62	0.000800142	RMS	RMS	MATCHED
Sample 63	2.77E-11	RMS	RMS	MATCHED
	Distance	Network Diagnosis	Real Diagnosis	Diagnosis Result
Sample 64	0.182286	NB	NB	MATCHED
Sample 65	0.161505	NOISE	EWS	No Matched
Sample 66	304.061	NOISE	NOISE	MATCHED
Sample 67	9.00E-10	RMS	RMS	MATCHED
Sample 68	31.5552	NOISE	NOISE	MATCHED
Sample 69	2.14E-06	EWS	EWS	MATCHED
Sample 70	0.00285675	BL	BL	MATCHED
Sample 71	0.00081042	NB	NB	MATCHED
Sample 72	2610.99	NOISE	NOISE	MATCHED
Sample 73	3.06E-08	RMS	RMS	MATCHED
Sample 74	495.435	NOISE	NOISE	MATCHED
Sample 75	0.0113069	EWS	EWS	MATCHED
Sample 76	3705.4	NOISE	NOISE	MATCHED
Sample 77	0.100186	NB	NB	MATCHED
Sample 78	0.00340232	BL	BL	MATCHED
Sample 79	0.000456761	NB	NB	MATCHED
Sample 80	1.21E-05	RMS	RMS	MATCHED
Sample 81	0.698487	NOISE	BL	No Matched
Sample 82	0.000336146	EWS	EWS	MATCHED
Sample 83	1078.04	NOISE	EWS	No Matched
Sample 84	0.211844	NOISE	EWS	No Matched
Sample 85	0.000600993	RMS	RMS	MATCHED
Sample 86	0.00020763	NB	NB	MATCHED
Sample 87	1.13E-07	RMS	RMS	MATCHED
Sample 88	0.00128197	NB	NB	MATCHED

Table 3: Diagnosis Result: the SRBCT dataset with 43 genes from Tibshirani et al., (2001) study is diagnosed with 95.4% accuracy.

	Real Output	Network Output	Classification	Network Outputs			
Sample 1	0	0	Matched	0.996814	0.002321	0.000265	0.003934
Sample 2	0	0	Matched	0.996834	0.002309	0.000264	0.003913
Sample 3	0	0	Matched	0.996839	0.002319	0.000259	0.003915
Sample 4	0	0	Matched	0.996844	0.002309	0.000256	0.003924
Sample 5	0	0	Matched	0.996816	0.002329	0.000262	0.003936
Sample 6	0	0	Matched	0.996812	0.00231	0.000268	0.003933
Sample 7	0	0	Matched	0.995448	0.002509	0.000369	0.005746
Sample 8	0	0	Matched	0.99674	0.00224	0.0003	0.003961
Sample 9	0	0	Matched	0.996784	0.002306	0.000275	0.003953
Sample 10	0	0	Matched	0.990821	0.00621	0.000317	0.010853
Sample 11	0	0	Matched	0.996853	0.00232	0.000253	0.003907

Sample	12	0	0	Matched	0.996822	0.002321	0.000264	0.003925
Sample	13	0	0	Matched	0.996688	0.002258	0.000297	0.00404
Sample	14	0	0	Matched	0.996541	0.002249	0.000362	0.003978
Sample	15	0	0	Matched	0.996706	0.002249	0.000318	0.003928
Sample	16	0	0	Matched	0.996515	0.001922	0.000442	0.00409
Sample	17	0	0	Matched	0.995737	0.001898	0.000387	0.005875
Sample	18	0	0	Matched	0.996564	0.00221	0.00036	0.003975
Sample	19	0	0	Matched	0.99646	0.00216	0.000384	0.004059
Sample	20	0	0	Matched	0.996817	0.002292	0.000274	0.003912
Sample	21	0	0	Matched	0.993007	0.00046	0.014874	0.003831
Sample	22	0	0	Matched	0.935942	0.004645	0.008998	0.006882
Sample	23	0	0	Matched	0.989371	0.005007	0.000569	0.007091
Sample	24	3	3	Matched	0.026632	0.009925	0.002573	0.975082
Sample	25	3	3	Matched	0.018321	0.004993	0.007077	0.979489
Sample	26	3	3	Matched	0.006012	0.015336	0.005776	0.985489
Sample	27	3	3	Matched	0.009541	0.00965	0.00491	0.986958
Sample	28	3	3	Matched	0.010033	0.012502	0.006387	0.978461
Sample	29	3	3	Matched	0.011458	0.007312	0.00636	0.98515
Sample	30	3	3	Matched	0.016902	0.00697	0.003456	0.986724
Sample	31	3	3	Matched	0.003736	0.014186	0.011615	0.986321
Sample	32	2	2	Matched	0.005568	0.00449	0.995073	0.006157
Sample	33	2	2	Matched	0.009945	0.00441	0.990494	0.005774
Sample	34	2	2	Matched	0.005142	0.004798	0.995273	0.006109
Sample	35	2	2	Matched	0.007587	0.004254	0.992763	0.006367
Sample	36	2	2	Matched	0.006839	0.004352	0.99459	0.005743
Sample	37	2	2	Matched	0.007405	0.011349	0.982771	0.004981
Sample	38	2	2	Matched	0.010568	0.003567	0.992531	0.005824
Sample	39	2	2	Matched	0.008387	0.004213	0.993296	0.005656
Sample	40	2	2	Matched	0.018135	0.005252	0.987673	0.00351
Sample	41	2	2	Matched	0.006588	0.004703	0.993329	0.006261
Sample	42	2	2	Matched	0.027902	0.002453	0.986505	0.005463
Sample	43	2	2	Matched	0.007426	0.006475	0.992758	0.004252
Sample	44	1	1	Matched	0.003574	0.995125	0.00342	0.005441
Sample	45	1	1	Matched	0.003676	0.995016	0.003249	0.00546
Sample	46	1	1	Matched	0.003581	0.995111	0.003369	0.005438
Sample	47	1	1	Matched	0.005459	0.987474	0.01119	0.003342
Sample	48	1	1	Matched	0.004212	0.994611	0.003155	0.005449
Sample	49	1	1	Matched	0.003817	0.99482	0.003595	0.00519
Sample	50	1	1	Matched	0.006623	0.991422	0.002457	0.007691
Sample	51	1	1	Matched	0.007856	0.990479	0.002269	0.007759
Sample	52	1	1	Matched	0.004835	0.993427	0.003175	0.00593
Sample	53	1	1	Matched	0.004123	0.994273	0.003232	0.005863
Sample	54	1	1	Matched	0.003708	0.995002	0.003314	0.00545
Sample	55	1	1	Matched	0.003825	0.994928	0.00318	0.00546
Sample	56	1	1	Matched	0.006968	0.993173	0.002499	0.005512
Sample	57	1	1	Matched	0.004332	0.994433	0.002978	0.005573
Sample	58	1	1	Matched	0.017415	0.978645	0.002199	0.007352
Sample	59	1	1	Matched	0.005062	0.993141	0.002858	0.006476
Sample	60	1	1	Matched	0.003589	0.995118	0.00332	0.005433
Sample	61	1	1	Matched	0.005364	0.990381	0.00474	0.005102
Sample	62	1	1	Matched	0.004323	0.993918	0.003403	0.005475
Sample	63	1	1	Matched	0.009504	0.989803	0.002701	0.004799
Sample	64	2	2	Matched	0.005116	0.004791	0.995238	0.006178
Sample	65	0	0	Matched	0.995306	0.002075	0.000506	0.00513
Sample	66	-1	-1	Matched	0.165492	0.003666	0.009118	0.776272
Sample	67	1	1	Matched	0.003602	0.995057	0.003364	0.005434
Sample	68	-1	0	No Matched	0.989069	0.008562	0.00036	0.009015
Sample	69	0	0	Matched	0.99685	0.002314	0.000255	0.003908
Sample	70	3	3	Matched	0.139182	0.002805	0.002289	0.983741
Sample	71	2	2	Matched	0.065118	0.004361	0.97696	0.00246
Sample	72	-1	-1	Matched!	0.107593	0.011737	0.004499	0.773208
Sample	73	1	0	No Matched	0.818622	0.065164	0.000413	0.070138
Sample	74	-1	-1	Matched!	0.020224	0.45701	0.005151	0.157963

Sample	75	0	0	Matched	0.996856	0.00232	0.000251	0.003906
Sample	76	-1	-1	Matched	0.087778	0.008677	0.002882	0.917461
Sample	77	2	2	Matched	0.009817	0.002731	0.992306	0.008478
Sample	78	3	3	Matched	0.017045	0.0063	0.004623	0.985604
Sample	79	2	2	Matched	0.006246	0.004328	0.99477	0.006096
Sample	80	1	1	Matched	0.003582	0.994977	0.003587	0.00542
Sample	81	3	3	Matched	0.008843	0.008212	0.00654	0.9874
Sample	82	0	0	Matched	0.99685	0.00232	0.000255	0.003909
Sample	83	0	3	No Matched	0.067289	0.118475	0.002468	0.468791
Sample	84	0	0	Matched	0.990644	0.006291	0.000319	0.011044
Sample	85	1	1	Matched	0.00363	0.995086	0.00324	0.005427
Sample	86	2	2	Matched	0.011302	0.002979	0.991996	0.007044
Sample	87	1	1	Matched	0.00383	0.994877	0.003153	0.005516
Sample	88	2	2	Matched	0.005937	0.004359	0.994964	0.006188

Table 4: Classification result, by using SRCT dataset with 7 genes from Pal et al.,(2007) study. This dataset is used on Multilayer Neural Network and the classification is resulted in 97.7% accuracy.

			Diagnosis	Diagnosis	Result
Sample	1	0.388625	EWS	EWS	MATCHED
Sample	2	0.384804	EWS	EWS	MATCHED
Sample	3	0.381141	EWS	EWS	MATCHED
Sample	4	0.376803	EWS	EWS	MATCHED
Sample	5	0.386558	EWS	EWS	MATCHED
Sample	6	0.390694	EWS	EWS	MATCHED
Sample	7	0.58031	EWS	EWS	MATCHED
Sample	8	0.410624	EWS	EWS	MATCHED
Sample	9	0.397596	EWS	EWS	MATCHED
Sample	10	0.854101	EWS	EWS	MATCHED
Sample	11	0.374333	EWS	EWS	MATCHED
Sample	12	0.38708	EWS	EWS	MATCHED
Sample	13	0.408077	EWS	EWS	MATCHED
Sample	14	0.452754	EWS	EWS	MATCHED
Sample	15	0.428048	EWS	EWS	MATCHED
Sample	16	0.464219	EWS	EWS	MATCHED
Sample	17	0.46353	EWS	EWS	MATCHED
Sample	18	0.457797	EWS	EWS	MATCHED
Sample	19	0.472322	EWS	EWS	MATCHED
Sample	20	0.392583	EWS	EWS	MATCHED
Sample	21	1.49421	EWS	EWS	MATCHED
Sample	22	10.5997	EWS	EWS	MATCHED
Sample	23	1.26975	EWS	EWS	MATCHED
Sample	24	5.57074	BL	BL	MATCHED
Sample	25	4.08905	BL	BL	MATCHED
Sample	26	3.73232	BL	BL	MATCHED
Sample	27	3.40453	BL	BL	MATCHED
Sample	28	3.9487	BL	BL	MATCHED
Sample	29	3.52176	BL	BL	MATCHED
Sample	30	3.98311	BL	BL	MATCHED
Sample	31	3.5523	BL	BL	MATCHED
Sample	32	0.387964	NB	NB	MATCHED
Sample	33	0.79381	NB	NB	MATCHED
Sample	34	0.34609	NB	NB	MATCHED
Sample	35	0.611812	NB	NB	MATCHED
Sample	36	0.433219	NB	NB	MATCHED
Sample	37	1.4296	NB	NB	MATCHED
Sample	38	0.656074	NB	NB	MATCHED
Sample	39	0.48204	NB	NB	MATCHED
Sample	40	0.901303	NB	NB	MATCHED
Sample	41	0.519648	NB	NB	MATCHED
Sample	42	1.58025	NB	NB	MATCHED

Sample 43	0.505193	NB	NB	MATCHED
Sample 44	0.705105	RMS	RMS	MATCHED
Sample 45	0.659873	RMS	RMS	MATCHED
Sample 46	0.682418	RMS	RMS	MATCHED
Sample 47	1.43015	RMS	RMS	MATCHED
Sample 48	0.690635	RMS	RMS	MATCHED
Sample 49	0.77066	RMS	RMS	MATCHED
Sample 50	0.778246	RMS	RMS	MATCHED
Sample 51	0.805794	RMS	RMS	MATCHED
Sample 52	0.784169	RMS	RMS	MATCHED
Sample 53	0.727093	RMS	RMS	MATCHED
Sample 54	0.684445	RMS	RMS	MATCHED
Sample 55	0.655431	RMS	RMS	MATCHED
Sample 56	0.694926	RMS	RMS	MATCHED
Sample 57	0.681097	RMS	RMS	MATCHED
Sample 58	1.78257	RMS	RMS	MATCHED
Sample 59	0.760929	RMS	RMS	MATCHED
Sample 60	0.661035	RMS	RMS	MATCHED
Sample 61	0.932684	RMS	RMS	MATCHED
Sample 62	0.740913	RMS	RMS	MATCHED
Sample 63	0.984347	RMS	RMS	MATCHED
	Distance	Network Diagnosis	Real Diagnosis	Diagnosis Result
Sample 64	1.75277	NB	NB	MATCHED
Sample 65	1.64822	EWS	EWS	MATCHED
Sample 66	1191.3	NOISE	NOISE	MATCHED
Sample 67	2.40363	RMS	RMS	MATCHED
Sample 68	1839.12	NOISE	NOISE	MATCHED
Sample 69	0.869795	EWS	EWS	MATCHED
Sample 70	46.558	NOISE	BL	No Matched
Sample 71	13.1437	NOISE	NB	No Matched
Sample 72	1154.92	NOISE	NOISE	MATCHED
Sample 73	2913.35	NOISE	RMS	No Matched
Sample 74	443.98	NOISE	NOISE	MATCHED
Sample 75	0.857647	EWS	EWS	MATCHED
Sample 76	1606.55	NOISE	NOISE	MATCHED
Sample 77	2.87532	NB	NB	MATCHED
Sample 78	11.0218	BL	BL	MATCHED
Sample 79	2.04084	NB	NB	MATCHED
Sample 80	2.50392	RMS	RMS	MATCHED
Sample 81	10.1616	BL	BL	MATCHED
Sample 82	0.872569	EWS	EWS	MATCHED
Sample 83	2073.01	NOISE	EWS	No Matched
Sample 84	2.52033	EWS	EWS	MATCHED
Sample 85	2.32747	RMS	RMS	MATCHED
Sample 86	2.62351	NB	NB	MATCHED
Sample 87	2.3921	RMS	RMS	MATCHED
Sample 88	1.81133	NB	NB	MATCHED

Table 5: Diagnosis result, by using SRCT dataset with 7 genes from Pal et al.,(2007) study. This dataset is used on Multilayer Neural Network and the diagnosis is resulted in 95.4% accuracy