UNIVERSITY OF VAASA

FACULTY OF TECHNOLOGY

COMMUNICATIONS AND SYSTEMS ENGINEERING

Alabi Rasheed Omobolaji

PREDICTION OF RECURRENCE AND MORTALITY OF ORAL TONGUE CANCER USING ARTIFICIAL NEURAL NETWORK

(A case study of 5 hospitals in Finland and 1 hospital from Sao Paulo, Brazil)

Master's thesis for the degree of Master of Science in Technology submitted for inspection, In Vaasa, 12.08.2017

Thesis Instructor	Dr Alhadi Almangush
Thesis Supervisor	Professor Mohammed Elmusrati

ACKNOWLEDGMENT

In the name Allaah the Most beneficent, the Most Merciful. All thanks to Allaah and I beseech His peace and benedictions on the noblest of mankind, Muhammad, peace and blessings of Allaah be upon him and the generality of Muslims till the day of accountability.

First and foremost, I would like to express my profound appreciation to my supervisor, Professor Mohammed Elmusrati, for his guidance throughout the development of this work and also throughout my Master's program. You are indeed a role model. I have gained both academic knowledge and knowledge towards a unique approach to life and situations from you. It is a rare opportunity to work with you and I really appreciate the opportunity. Without an iota of doubt, you have left a positive mark in my life and I will always remain grateful. I am very proud to be your student.

Similarly, I sincerely appreciate the efforts, contributions and continuous monitoring of the progress of the work from my instructor, Dr Alhadi Almangush. It was indeed a good learning curve to have worked with you. The professionalism and maturity shown during the course of this work was well appreciated. Most importantly, the opportunity given to me to work the dataset of the cancer patients. I thank you for the guidance throughout this work. I will forever be grateful to you.

Furthermore, my deepest appreciation goes to my beloved mother for her unconditional love and unending support right from my primary school days. My mother is indeed the best teacher that I ever know. Talking to her alone gives me the joy and happiness needed to continue with my day to day activities.

The whole of my Master program (MSc) and this thesis is specially and lovely dedicated to my wonderful wife, Ummu Mu'adh - Atunrase Mistura. She encouraged me to pursue this MSc program. Her emotional support was vital while acknowledging the long weekly separation and discomfort we both endured to accomplish this fit. It is not easy travelling from Helsinki-Vaasa-Helsinki on a weekly basis . She believed in me more than I have believed in myself and that gave me the courage to persevere whenever I hit roadblocks. I love you so much.

To my lovely son, Mu'adh Adebayo, May Allaah bless you. Thinking about you alone is enough for me to be happy. I love you so much. It was not easy to leave you in Helsinki on a weekly basis while I was busy in Vaasa with my studies. I sincerely appreciate the understanding. I pray to Allaah to make you a scholar of high repute.

I am thankful to Professor Timo Mantere, Tobias Glocker, Ahmed Elgrgouri, Dr Ali Altowati, Shaima AbdulMageed for their immense contribution in deepening my knowledge through the various courses taught in this masters degree program. This acknowledgement will not be complete without showing appreciation to my brother and friend, AbdulRahman Olaobaju for his understanding and numerous tutorials to make sure I meet up with my academics. To all the members of academic and non-academic staff of the Faculty of Technology, Communications and Systems Engineering Group especially Marjukka Isaksen, I say a big thank you for their contribution to the success of this program. My classmates must be acknowledged for their support and positive contribution during the course work.

Vaasa, Finland, August, 2017,

Alabi Rasheed Omobolaji

CONTENTS

SYMBOLS

ε (t)	Error of approximation of the series
Θ_i	Threshold of the unit i.
a_1.....a_n	Inputs
b_j	Thresholds
D^i_1....D^i_N	Delays
f(x)	Sigmoid function
f'(x)	Sigmoid function for back propagation
h()	Previous Values
$I^1(t)$ .....$I^M(t)$	Inputs
N_1....N_M	Number of Delays
n	Number of data sample
net	Neural Network
O_j	Output
S_j	State of Unit J
T1	Tumour Size one
T2	Tumour Size two
trainFcn	Training Function
u	Weights
v	Weights
W_{nj}	Weights
x(k)	Outputs of the hidden layer
$x_c(k)$	Outputs of context layer
ý	Approximated data obtained by the network for value y_i

$\mathbf{y_i}$	i-th Data Sample
$\mathbf{y_j(t)}$	Output of hidden layer
$\mathbf{y_k(t)}$	Final Output
y(t)	Data series for prediction
y(t-1)..y(t-p)	Past Values / Feedback Delays

ABBREVIATIONS

AI	Artificial Intelligence.
ANN	Artificial Neural Network.
BAM	Bidirectional Associated Memory (BAM)
BPTS	Back Propagation Through Structure
BPTT	Back Propagation Through Time
CAFs	Cancer Associated Fibroblasts
cTNM	Cancer Tumour Size, Lymph Nodes, Metastasis.
DNA	Deoxyribo Nucleic Acid
dividerand	This function divides the dataset automatically
ENN	Elman Neural Network.
Elmannet	Elman Neural Network function
feedforwardnet	Feedforward Neural Network function
FS	Feature/Input Selection
GA	Genetic Algorithm
HNN	Hopfield Neural Network
ICT	Information Communication Technology
layrecnet	Layer Recurrent Neural Network function
LHR	Lymphocytic Host Response
LMBP	Levenberg-Marquardt Propagation Procedure
LRNN	Layer Recurrent Neural Network
LVQ	Learning Vector Quantization
RL	Reinforcement Learning
RNN	Recurrent Neural Network

NAR	Nonlinear Auto Regressive
NN	Neural Network.
NARNET	Nonlinear Autoregressive Neural Network function
NARXNET	Nonlinear Autoregressive Neural Network with external input function
MATLAB	MathWorks Simulation Tool R2014b.
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
PNI	Peri Neural Invasion (PNI)
SCC	Squamous Cell Carcinoma.
SGD	Stochastic Gradient Descent
SMA	Smooth Muscle Acting
SSE	Sum of Squares Error
SVM	Support Vector Machine.
tr.trainInd	This function divides the dataset into training set
tr.valInd	This function divides the dataset into validation set
tr.testind	This function divides the dataset into testing set
trainFcn	Training Function
timedelaynet	Time Delay Network
TDN	Time Delay Neuron
TDNN	Time Delay Neural Network
VALVIRA	National Supervisory Authority for Welfare and Health.
WPOI	Worst Pattern Of Invasion

LIST OF FIGURES **Page**

LIST OF TABLES **Page**

UNIVERSITY OF VAASA
Faculty of technology
Author: Alabi Rasheed Omobolaji
Topic of the Thesis: Prediction of Recurrence and Mortality of Oral Tongue-Caner using Artificial Neural Network
Supervisor: Professor Mohammed Salem Elmusrati
Instructor: PhD Alhadi Almangush
Degree: Master of Science in Technology
Major of Subject: Communication and Systems Engineering
Year of Entering the University: 2015
Year of Completing the Thesis: 2017 **Pages: 180**

ABSTRACT

Cancer is a dreadful disease that had caused the death of millions of people. It is characterized by an uncontrollable growth of cell to form lumps or masses of tissue that are known as tumour. Therefore, it is a concern to all and sundry as these tumours mostly release hormones which have negative impact on the body system. Data mining approaches, statistical methods and machine learning algorithms have been proposed for effective cancer data classification. Artificial Neural Networks (ANN) have been used in this thesis for the prediction of recurrence and mortality of oral tongue cancer in patients. Similarly, ANN was also used to examine the diagnostic and prognostic factors. This was aimed at determining which of these diagnostic and prognostics factors had influence on the prediction of recurrence and mortality of oral tongue cancer in patients. Three different ANN have been applied for the learning and testing phases. The aim was to find the most effective technique. They are Elman, Feedforward, and Layer Recurrent neural networks techniques. Elman neural network was not able to make acceptable prediction of the recurrence or the mortality of tongue cancer based on the data. In contrast, Feedforward neural network captured the relationship between the prognostic factors and correctly predicted recurrence. However, it failed to predict the mortality based on the patient's data. Layer Recurrence neural network has been very effective and successfully predicted the recurrence and the mortality of oral tongue cancer in patients. The constructed layered recurrence neural network has been used to investigate the correlation between the prognostic factors. It was found that out of 11 prognostic factors in the data sheet, it was only 5 of them that had considerable impact on the recurrence and mortality. These are grade, depth, budding, modified stage, and gender. Time in months and disease free months were also used to train the network.

KEYWORDS: Artificial Neural Network, Feedforward, Elman, Layer Recurrent, Recurrence, Mortality, Prediction, Prognostic factors, Cancer, Oral Tongue

1 INTRODUCTION

Cancer is a distressing, alarming and dreadful disease. Several death cases have been recorded as a result of this atrocious disease. The figure of cancer-related death makes cancer a concern to the medical practitioner. Cancer is one of the main causes of death in many developed nations. Similarly, in developing countries, the impact of cancer, as well as diabetes as major players in the death rate, cannot be over emphasized (Shikha & Jitendra 2015) . It's very important for treatment of cancer to classify tumor accurately. As with other diseases, proper identification, classification, and prediction are some factors to achieving efficient and effective treatment and management for cancer patients. Therefore, proper identification of cancer cells is ultimately an important step. In developed countries with advanced and up to date medical facilities, ineffective cancer classification methodology has been a major cause of death due to cancer-related condition because cancer classification was medically based on clinical and histopathological facts.

More often than none, this classification approach often produces incomplete or misleading results. Thus, the need to consider other options in the classification of cancer in patients becomes imperative. DNA microarray, molecular level diagnostics to mention but a few offered a way of cancer classification (Oliver et al 2009:157; Wang et al 2005). Although, both DNA microarray and molecular level provide accurate prediction and diagnosis of cancer. Furthermore, gene expression data generally comprise of a huge number of genes has been a major concern for the stakeholders in the medical discipline. Hence, a better approach to efficient and effective classification and prediction through other disciplines and field of study becomes imperative.

Data mining approaches, statistical methods and machine learning algorithms to effectually evaluate these data have been proposed (Sung-Bae Cho & Hong-Hee 2007). As a result of this, support vector machine (SVM), k-nearest neighbor and neural network techniques to mention but a few have attracted more attention in recent years. Specifically, the use of Artificial Neural Networks (ANN) in medical research is on the increase in recent years. It has been extensively employed in medical areas such as urology, radiology, medical microbiology, medical biochemistry to mention but a few. The use of ANN in other medical areas as well as in the analysis of patient's data is at a geometric rate.

Neural network model has been successfully implemented in various classification problems. Neural network techniques are very useful for detection, prediction, and monitoring of cancer. For instance, it was recently used in the clustering and classification of gene expression data. There are two models used. These are the supervised and unsupervised models. Classification can be achieved with the supervised models while unsupervised models are used for clustering. Classification is crucially important for cancer diagnosis and treatment. The artificial neural network had been proven to be a very effective method for pattern recognition. This made them very useful for diagnosis of cancer disease at very early stages.

Oral tongue cancer is the most common and the most aggressive epithelial cancer diagnosed within the oral cavity. The incidence of oral tongue cancer has increased tremendously and thus it has attracted attention of the clinicians and researchers. Therefore, the focus of this thesis is to examine the use of ANN in oral tongue cancer recurrence and mortality prediction. In addition, the thesis will examine ANN as a means to determine which of the prognostic factors are needed in the prediction of recurrence and mortality. Though, classification, as used in medicals includes detection, prediction, and treatment based on certain parameters. For the purpose of this thesis work, much attention will be given to prediction based on the learning outcome of MATLAB ANN using the dataset provided, that is, the case study. In addition, fixed and dynamic dataset generated by the supervisor based on certain algorithm would be tested to see the effectiveness of the code prior to testing with the real data, that is, the case study data.

Furthermore, this thesis would also attempt to examine if *modified_stage* is a good prognostic factor to be considered in the prediction of recurrence and mortality. These can be achieved by removing certain columns from the given dataset during the training in MATLAB ANN to see if such column has any effect on the expected outcome. Conclusively, the thesis is poised to be instrumental as an approach for clinicians to achieving an efficient way to the prediction of the patient's situation. The description of the dataset can be found in **Section 1.1** named as the dataset. The motivation gives further explanation to the thesis and it is worthy of note that the thesis will focus on tongue cancer patients.

1.1 The Dataset

This research entails the use of data from patients. Therefore, the ethical and privacy concern is put into serious consideration. The use of patient data were by the National Supervisory Authority for Welfare and Health (VALVIRA) (Almangush et al 2013). Despite this approval, it is worthy of note that the identification number used in this data set for each patient has been coined and developed by the author of this research work. This measure is to enhance and protect the privacy further as this research work will be accessed publicly (online or printed means) through the University of Vaasa. Hence, the identification given here has no relationship with the identification number as contained in the original data. The diagnostic histological slides of 340 patients with T1/T2 N0M0 oral tongue Squamous Cell Carcinoma (SCC) managed between 1979 and 2009 from the University Hospitals of Helsinki, Oulu, Turku, Tampere, and Kuopio were collected from the hospitals' archives. Similarly, the data of patients from one hospital in San Paulo in Brazil were also included. The criteria for inclusion of cases were as described in (Brandwein-Gensler et al 2010).

1.2 Motivation

Undoubtedly, early detection of any disease or cancer to be precise gives a good insight into the disease and ultimately, the management practicalities of such disease. In cancer detection and prediction, machine learning such as Artificial Neural Network is one of the methods being investigated. ANNs offered a unique and efficient approach to cancer prognosis due to ANN's ability to learn and generalize from data. The dataset as contained in Appendix III was used for analysis in this thesis. The data was supplied by the Dentistry Department of the Helsinki Teaching Hospital. The data has been named as Appendix III. Thus, this thesis is aimed to examine and identify the optimal cutoff point of tumor depth (column F) for risk stratification in T1 and T2 stages separately; and for both stages together. Furthermore, the thesis is poised to determine if the tumor depth is used to modify cTNM staging system (column K) with the help of ANN? In addition, is the modified T-stage (column G) is better in prognostication? Finally, this research work will aim to identify the optimal cutoff point of tumor budding (B) for risk stratification in T1 and T2 stages separately; and for both stages together. For future study, it would be ideal to test and examine the interpretation of ANNs for the other prognostic factor (e.g. WPOI, Grade.... etc) as contained in the data set.

1.3 Thesis Structure

The thesis is organized in six chapters, Chapter one deals with the introduction of the subject areas of the research and the research questions. The literature review on the previous work on Artificial Neural Network for cancer prediction and diagnosis is examined. An overview of the Jeff Elman Neural Network is presented. Elman Neural Network (ENN) has been chosen to be the neural network in the prognostication of mobile cancer because of his ready-made function for the medical application. The recurrent nature also made it a good choice for this research work. Although there are better NN than Elman such as layer recurrent that produces better performance than Elman.

Therefore both Elman and layer recurrent will be mainly examined. All these would be contained in **Chapter 2**. The third (3) chapter presents the recent application ANN in medicines. **Chapter 4** examines simulations of the ***Feedforward*** and ***Elman*** neural network approach using some arbitrary data. The data was supplied by my thesis supervisor and generated based on a certain algorithm. The aim of this chapter is to understand the difference between static and dynamic variables. The case study would be examined in **Chapter 5**. Comprehensive analysis of the data set using the ANN so as to answer the research questions presented in **Chapter 1**. **Chapter six (6)** involves discussion on the results obtained.

The research questions presented in **Chapter 1** forms the foundation of this research question. In addition, the future research question presented in chapter 1 will be equally examined. The main contribution of this thesis is in **Chapter 6** where conclusions, recommendations, and possibility for future study are made based on the results presented in **Chapter 5**. The Appendix I-III contains few samples of the dataset and Appendix IV contains some of the useful codes for the ANN.

2 FUNDAMENTALS OF ARTIFICIAL NEURAL NETWORKS

2.1 Artificial Neural Network

An Artificial Neural Network (ANN) is a statically oriented modeling tool. It is a similitude of the biological nervous system. The basic processing element in the ANN is known as the neurons. The neuron is not the same as the neuron in the human body but in terms of functionality, it works in the same manner. Hence, the name artificial neurons. It has a normal range of output between (-1, +1). It could also be (0, 1) (Ying et al 1998, Ferari & Stengel 2005). The neuron can be viewed as a processor that computes the sum of weighted inputs and then applies a non-linear transfer function to the computed sum. The example of transfer function could be Tang-sigmoid. **Figure 1** shows an example of artificial neuron and its components.

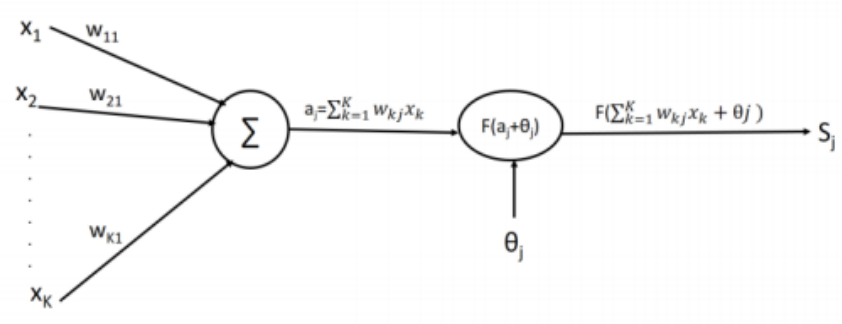


Figure 1. Artificial neuron and its components (Hassan et al 2016).

From the structure of the components of artificial neurons depicted in **Figure 1**, it can be said that an artificial neuron consists of inputs and weights, a transfer function and an activation function (Heimes & B.van Heuveln 2005, Jayadeva et al 2002, Setino & A.Gavada 2000). These neurons are interconnected to each other for the purpose of working in unison to address a particular problem. In most fields of study, it is becoming imperative to detect trends and extract the patterns in some scenarios. Doing these actions with the traditional method, that is, through human and statistical and computer techniques are becoming increasingly difficult.

Therefore, ANN has come to provide a unique approach to solving the problems. This unique characteristic of the ANN has it to be widely used in so many applications nowadays-engineering, medicine, statistics to mention but a few. Neural network operates in a similar way as an adaptive system. By that, it means that it changes its structure during the learning phase. ANN has been touted to be effective in modeling simple and complex relationships. With regards to its application in data science, it can be used to find out patterns and clusters in data (Spelt et al 2013).

Today, ANN represents a major extension to computation. It provides better results and performance than the traditional statistical tools for the prediction and classification purposes in various applications (Paliwal M. & Kumar U.A 2009). ANNs offers short computation times, low computational burden and the opportunity of reformulating the problem thereby considering only on the important variables and parameters from the given data set or certain unknown areas of interest. Different types of neural networks are designed and developed for various applications; however, the solution offered is yet to reach 100% accuracy but the contribution cannot be over emphasized.

Artificial Neural Networks (ANN) based expert system in tongue cancer study has been attracting much attention in the recent years just as the use of ANN had gained popularity amongst researchers in automatic breast cancer diagnosis in the past few years (Ubeyli 2007, Karabatak & Ince 2009, Furundzic et al 1998). Most of the current approach to the cancer diagnosis and treatment had been based mostly on the years of experience of the medical officers. ANN is however poised to approximate complex and non-linear problems without having to know the mathematical representations of the system or learn from the wealth of experience of the medical officer.

This exemplary feature has made ANN to attract attention in the study of cancer, especially in cancer case prediction. Clinical sizes (T1 or T2) of early oral tongue cancer had failed to differentiate between patients with the possibly favourable condition and patient with the adverse outcome when both are given treatments respectively (Kellermann et al 2007:849-853).

In addition, the early stage detection (T1/T2N0M0) of tongue cancer does not always represent a vibrant and viable prediction of oral cancer as 20-40% already have spread to other areas (metastasis) at the presentation stage (Ganly et al 2012, Ho et al 1992).

Therefore, the need to have an effective and efficient ways of prediction becomes imperative. The prediction at the early stage of oral tongue cancer is necessary because it gives the opportunity to identify subsets of patients that have the probability of unfavourable condition from the mobile tongue cancer. Thus, such patients will need more aggressive treatment. Modality therapy is a good approach for such category of patients. Conversely, the prediction result would also provide the opportunity to know the patient's subsets that have the chances of a favourable outcome. In such case, surgical treatment should suffice for this latter case (Kellermann et al 2007:849-853).

Hence, to differentiate between the two cases provided above, ANN has been touted to be a reasonable approach. Therefore, the idea is to supply the neural network with sufficient training data, and subsequently find relationships between these data without requiring user intervention. For example, in this thesis research, the status of the patients can be inferred from the results obtained as presented in **Chapter 5**. However, designing an ANN is a complex task. Various design aspects and parameters such as choosing an optimal network topology, suitable learning algorithm, the initial value of the weights, learning rate to mention but a few needs to be optimized properly to enhance the efficient performance of the ANN. It is imperative to mention that network topology includes the number of hidden layers and nodes. Also, some researches and books also considered input/feature selection (FS) as part of the ANN design (Walczak & Cerpa 1999).

There is evidence that the FS has significantly improved ANN performance (Setiono & Liu 1997). Similarly, the design parameters and input feature subsets of ANN needs to be optimized. This is because the duo are problems that have an influence on each other. ANN offers some unique features. The learning process as an important feature of ANN has given flexibility to ANN in terms of its application.

For example, ANN could be used for data classification and pattern categorization through a learning process. Neurons are arranged to form layers and connection pattern. Based on these arrangements, different network configurations and structures can be formed.

Based on these, ANN can be divided into feed-forward network and recurrent network. Feed-forward was considered to be the first and simplest artificial neural network. By feed-forward network, it means that: (i) the neurons in the network can be ordered without having a backward connection, that is, independent of time (ii) the output does not depend on the past (not cyclic) (iii) movement is in the forward direction (input through hidden nodes to the output nodes).

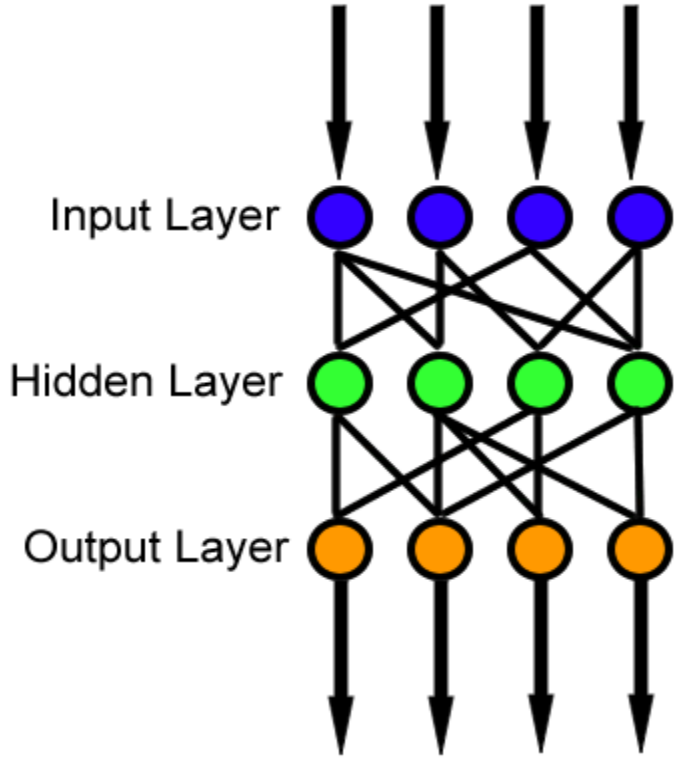


Figure 2. Feed-forward network structure (Tahmasebi et al 2011).

The feed-forward network is characterized by neither cycles nor loops in the network. It has been used extensively in classification, pattern recognition, and prediction. Conversely, the recurrent network contains loops and it has been used for processing tasks and control signals (Turkson et al 2016). An example of recurrent network is the Elman network. **Figure 3** shows the diagrammatic representation of the concept of Elman neural network.

As shown in **Figure 3**, Elman is made up of three-layers network. These layers could be arranged to form cluster of layers each containing these three-layers. They are represented as A, B and C in **Figure 3** respectively. The B layer is referred to as the hidden layer. The input are fed into the system in the forward direction through the first layer. That is why Elman neural network is also a variant of feedforward neural network. Apart from these three-layers, there are also the context units represented as K in Figure 3.

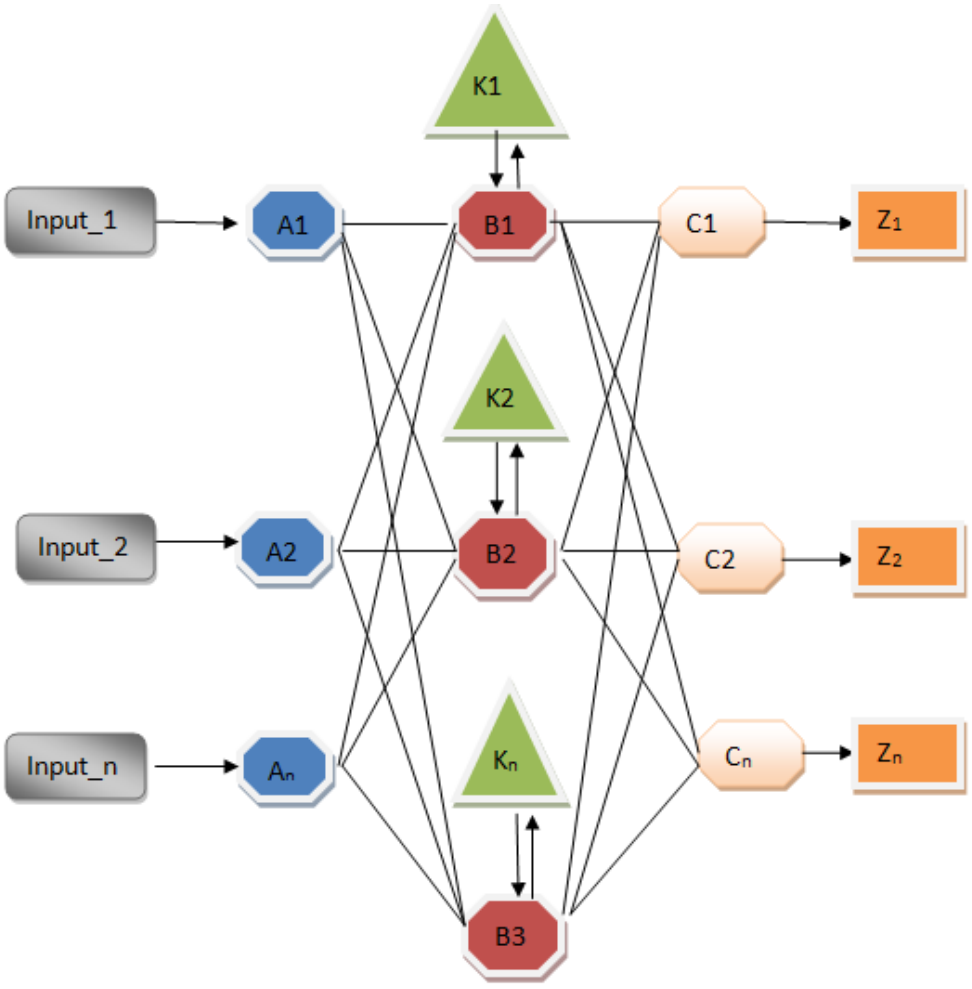


Figure 3. The concept of Elman neural network .

The hidden layer basically serves two functions. It connects to the context layer with a weight of one. Additionally, a copy of the previous value of the hidden units is stored in the context

units through back connections. This memory characteristic allows the network to remember the previously hidden layer states.

When the inputs are feed-forwarded into the system, an appropriate training algorithm would be applied and thus, the learning process could be completed. Where the value (B* W_1) is between the first and second layer and (C * W1) is the value fed into the third layer respectively (Elman 1990).

Due to the complicated design issue and coupled with the requirement for the FS, there is an increasing suggestion of hybridizing the ANN design with the evolutionary algorithm.
For instance, design methods such as trial-and-error, cannot simultaneously handle many design parameters and FS as well. Hence, the need for hybridization with evolutionary algorithm becomes imperative. Genetic Algorithm (GA) has global search features that makes it to be particularly preferred. GA is capable of generating both optimal feature subset and support vector machine (SVM) parameter without degrading the accuracy of the machine (Rui et al 2005; Huang & Wang 2006).

Hence, there have been several efforts that combine GA with ANN. For instance, GA was used to search for the architecture, learning algorithm and nodes' activation function (Ferentinos 2005), while the search for learning algorithm together with their parameters, hidden layer information, transfer function, weights and biases values was also an option (Almeida & Ludermir 2010). Additionally, a hybrid technique that combined fuzzy clustering, statistical tool and granular computing have been proposed (Yuchun et al 2008). A comprehensive review on combining the evolutionary algorithm with the ANN can be found in (Yao 1999). Promising results were obtained from combining evolutionary algorithm with ANN. Therefore, this thesis is aimed to further explore the ANN for mobile (oral) cancer prognosis. It employs the given dataset for FS, initial weight and hidden node size optimization of the most common ANN architecture. The dataset (Appendices I-III) were divided into training, validation and testing datasets respectively for the simulation exercises presented in **Chapter 4** and **Chapter 5** of this thesis write-up.

2.2 Classification of ANN

ANN can broadly be classified as either feed-forward and recurrent ANN respectively. An example of the feed-forward network includes:

I. Single layer feed-forward network.
II. Multilayer feed- forward network.
III. A Single node with its own feedback.

Similarly, an example of recurrent layer includes:

I. Multilayer recurrent network

A single layer feed-forward structure is a simple perceptron. The schematic diagram is as shown in Figure

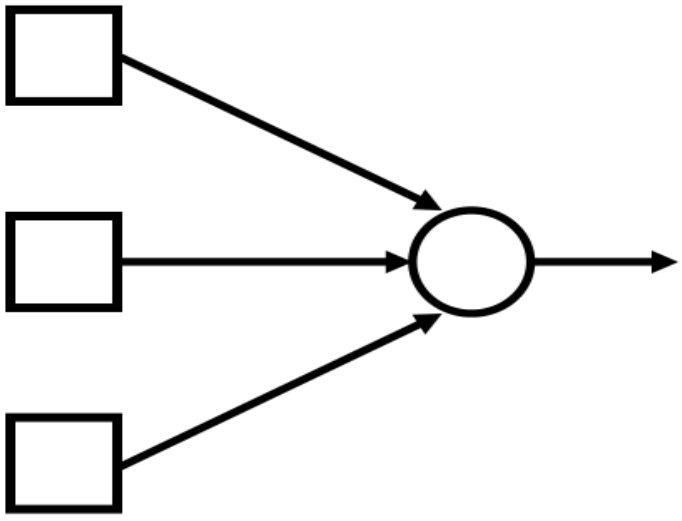


Figure 4. Single layer feed-forward network.

A Single-layer feed-forward network as the name implies has one input layer, one output layer, and no feedback connections. Inputs are applied to the network and with the aid of series of weights, and subsequently to the outputs. Inputs are multiplied by the weights in each node and compare with a threshold as shown in **Figure 7** below.

The leverage for comparison is that if the value obtained from the product of inputs and corresponding weights are above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1). Neurons that exhibit such kind of behavior are called *artificial neurons* or *linear threshold units*. In the literature, the term *perceptron* often refers to networks consisting of just one of these units. A similar neuron was described by Warren McCulloch and Walter Pitts in the 1940s.

A perceptron can be created using any values for the activated and deactivated states as long as the threshold value lies between the two. Most perceptrons have outputs of 1 or -1 with a threshold of 0. Multi-layer neural network on the other is characterized by the fact that it can calculate continuous output instead of a step function. Sigmoid function or logistic function provides a common choice for the multi-layer neural network. The Sigmoid function is given by:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Furthermore, the fact that sigmoid function has a continuous derivative has made it a preferred way in ANN as it can be used extensively in back-propagation.

$$f'(x) = f(x)(1 - f(x)).$$

Interestingly, the derivative of the function can be easily calculated as depicted in the equation above. Multilayer recurrent neural network or simply recurrent neural network (RNN) is also examples of ANN. Multi-layer perception (MLP) is made up of two or more layers of neurons that are connected sequentially. The connection between neurons in the different layer is by weighted signal pathways.

Signals are sent through these pathways to the other neurons. The Input layer is the first layer of a network. It receives signals from the data entering the network. The last layer, called the output layer, generates the outcome to the outside world.

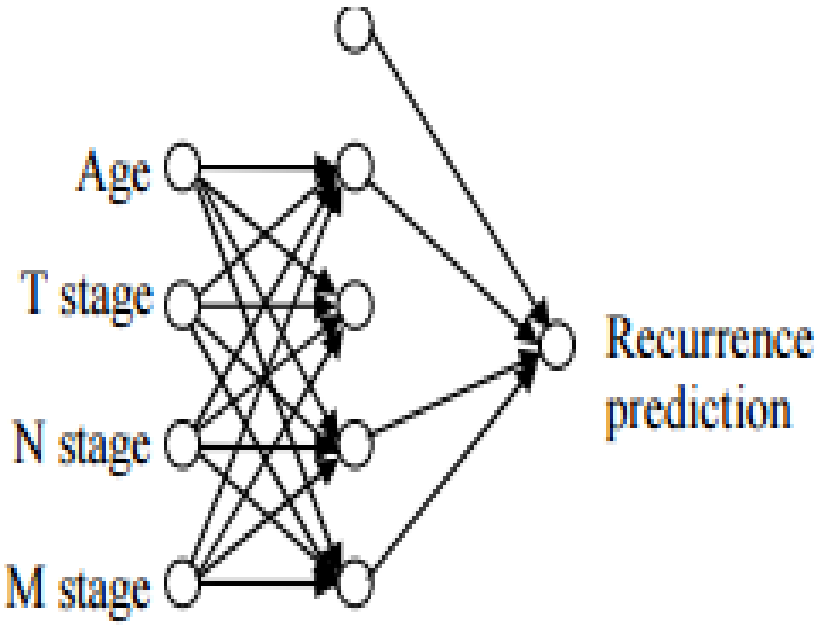


Figure 5. The structure of multi-layer perceptron.

Connections between various units of the network are in a fashioned and directed cycle. This fashioned and directed style gives it a dynamic temporal behaviour. Also, their internal memory can also be used to process arbitrary sequences of inputs. Thereby making RNN to be tremendously used in so many applications. Similarly, it uses back-propagation algorithm as well. However, it is worthy of note that back propagation is mainly used for networks that have activation functions that are differentiable. In addition, there are some issues that are associated with back-propagation. These include speed of convergence, over fitting and the possibility of ending up in a local minimum of the error function.

2.3 Training of ANN

Training or learning as it is otherwise called. ANN learning processes have been divided into three namely:

I. Supervised learning
II. Unsupervised learning
III. Reinforcement learning

When ANN is trained in the presence of an instructor, teacher or someone that is more knowledgeable about how ANN works, then such training process is termed as supervised learning process of ANN (Hu et al 1994). This type of training minimizes the possibility of error in the training process. This is because it is assumed that the pre determined target outputs values are known for each input pattern. Back propagation, time delay, multiple adaptive linear neurons to mention but a few are all examples of supervised neural networks.

Similarly, unsupervised training is characterized by the fact that it eschews the output knowledge from the instructor (Hu et al 1992;1994). In this case, the network finds the relationship between the inputs and the output by itself. Kohonen Self-Organizing Feature Maps and Learning Vector Quantization (LVQ) are examples of unsupervised training. Reinforcement learning (RL) is an area of machine learning inspired by behaviourist psychology. RL is beyond the scope of this thesis. By definition, training describes the procedure by which the parameters are tuned or adjusted in such way that makes the neural network to adapt itself to a stimulus. This parameter tuning consequently produced the desired output. The desired output is mostly compared with the expected output to see how effective the network had learned during the training period.

In general, neural network uses some internal calculations to computes output values from input values as shown in **Figure 6** (Delgrange et al 1998). The values of the weights are generally adjusted between the inputs, expected output and the target until the network produces meaningful results with the targets. Such output is known as the trained output or neural output. Thus, for any given set of inputs, the trained network can predict the correct outputs. Traditionally, not all the dataset are used for training to enhance the effectiveness of the output and most importantly, minimize errors between the expected output and the target output.

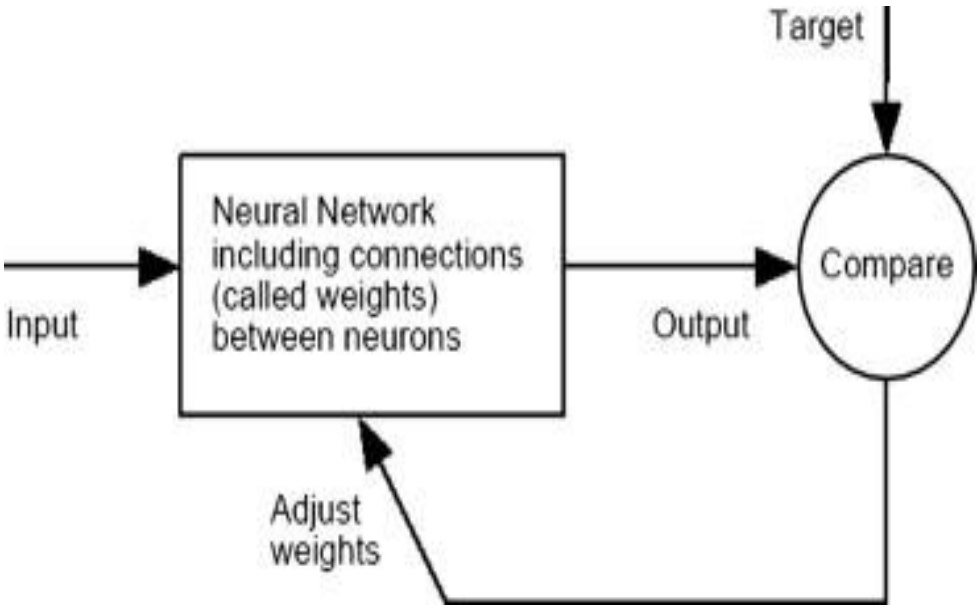


Figure 6. Neural Network training structure (Delgrange et al 1998)

Therefore, datasets are generally divided into training, validation and testing datasets respectively. Training dataset is essentially used to compute the gradient. Also, it is used to for updating the network weights and biases. The validation datasets are used for validation purpose. The training phase equally monitors the errors during the validation process. However, the testing datasets are not used at all in the training process. The testing dataset are used mainly for comparison purposes. The testing dataset provides important measures on how well the trained network has learned in the training phase.

2.4 Training Algorithm

Since the thesis would examine both feed-forward neural network and the recurrent neural network, it is pertinent to examine some of the training algorithms used. Some of the training algorithms are given in **Table 1**.

Table 1. Training Algorithms of ANN.

Training Function	Algorithms
trainlm	Levenberg-Marquardt
trainbr	Bayesian Regularization
trainbfg	BFGS Quasi-Newton
traingdm	Gradient Descent with Momentum
traingd	Gradient Descent
trainoss	One Step Secant

The training algorithms are numerous and each neural network types has a default training algorithm that is appropriate for it. A few of the training algorithms are presented in **Table 1**. The most widely used algorithms are Levenberg-Marquardt and Quasi-Newton methods because they are very fast and produce exceptional computational errors. Both are mostly used for datasets that are not much. For datasets that are large, Scaled Conjugate Gradient and Resilient Backpropagation are mostly the preferred options for training.

However, the default training method for feedforward network is the Levenberg-Marquardt. It is worthy of mention that the term backpropagation refers to gradient descent algorithm for the training of neural network (Demuth et al 2007).

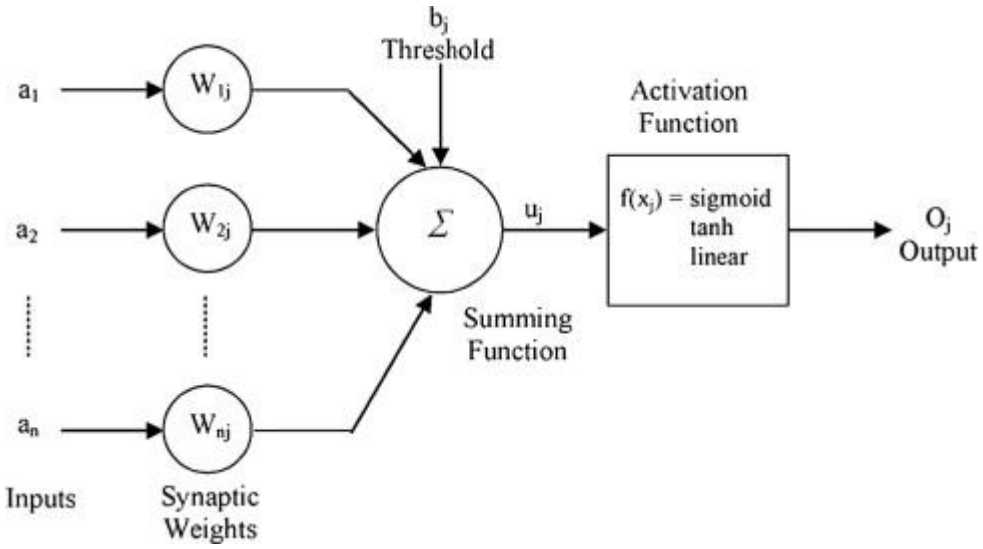


Figure 7. Single node anatomy of ANN.

As pointed out earlier, neuron or node as it can also be called formed the basic component of ANN. **Figure 7** showed the anatomy of a single node ANN. Where the inputs are a_1, a_2 and a_n, and the output by O_j. As shown in **Figure 7**, the node is the summing point. The node can accept more inputs than the ones shown in **Figure 7**. The function of the node is to manipulate the inputs to give a single output signal. The values W_{1j}, W_{2j}, and W_{nj}, are weight factors associated with the inputs to the node. Weights are adaptive coefficients within the network that determine the intensity of the input signal. Each input ($a_1, a_2, \dots, a_n$) is multiplied by its corresponding weight factor ($W_{1j}, W_{2j}, \dots, W_{nj}$), and the node uses summation of these weighted inputs ($W_{1j} * a_1, W_{2j} * a_2, \dots, W_{nj} * a_n$) to estimate an output signal using a transfer function.

The other input to the node, b_j, is the node's internal threshold, also called bias. This is a randomly chosen value that governs the node's net input through the following equation:

$$u_j = \sum_{i=1}^{n} (W_{ij} * a_i) + b_j$$

Node's output is determined using transfer function on the node's net input. Sigmoid, hyperbolic tangents and linear transfer functions can be effectively used. The transfer function can transform the node's net input in a linear or non-linear manner.

Sigmoid Transfer Function

$$f(x) = \frac{1}{1 + e^{-x}} \quad 0 \leq f(x) \leqslant 1$$

Hyperbolic Tangent Transfer Function

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad -1 \leqslant f(x) \leq 1$$

Linear Transfer Function

$$f(x) = x \quad -\infty < f(x) < +\infty$$

The neuron's output O_j is obtained based on any of the afore-mentioned transfer functions on the neuron's net input u_j. Hence, the equations above are transfer function equations that can be applied to the net input U_j to give the neurons' output O_j.

2.5 Advantages and Disadvantages of ANN

Advantages of ANN:

I. Adaptive learning: Sequel to the learning exercise for the network using appropriate training algorithm, the network can perform the task based on the data given for training.
II. Self-organization: After receiving the information in learning time an ANN can create its own organization.
III. Real-time operation: Many neural network computations can be carried out parallel. Specific hardware devices are being designed to take benefit of this ability of neural networks.
IV. Fault-tolerance via redundant information coding: Partial damage of a neural network structures lead to the degradation of performance. Though, some network abilities may be recollected even after major network damage.

Disadvantages of ANN:

I. Size and Complexity: Neural networks size and complexity is very high.

3 APPLICATION OF ANN IN MEDICINE

3.1 Artificial Neural Network in Medicine

The advancements in the field information and communication technology (ICT) have always been at a geometric rate. These advancements have been felt positively in other areas of endeavours. Medicine is not an exception. The tremendous development of ICT had contributed immensely to medicine through the development of powerful tools such as lasers, ultrasonic and so on that could aid medical treatments. Other areas that ICT have contributed to medicine are in data analysis and machine learning. For instance, Artificial Intelligence (AI) has contributed immensely to medicine and biological research. ANNs are an interesting and extensively studied branch of AI. It has been touted as a promising research area and it is opined by researchers in the field of machine learning and data science that ANNs would have extensive application to various biomedical problems in the future. Presently, it has gained the needed audience and attention in medicine as it was successfully applied to medical areas such as diagnostic systems, biochemical analysis, image analysis, and drug development (Konstantina 2017). This thesis will look at the application of ANN from the prediction of patient's situation point of view.

ANNs have been extensively applied in diagnosis, electronic signal analysis, medical image analysis, and radiology. ANNs is aimed to assist the doctors to detect the complex nonlinear relationships between dependent and independent variables in the patient's data. The neural network is able to learn, capture, draw inferences and establish a relationship from the provided data. This is always produced as an output of the learning process. Therefore, trained ANNs are the digitized model of the biological brain. Nowadays, ANNs are widely used for medical applications in various disciplines of medicine especially in cancer treatment, cardiology and so on.

Also, it has been used extensively in diagnostic systems because the ANN is not affected by other factors such as stress, fatigue, working conditions, emotional states, and equipment error and so on that could affect the traditional diagnostic procedures. The network can easily be trained and the trained network can produce an output that demonstrates that the network understood the relationship between the variables contained in the dataset given. Furthermore, ANNs have also found its application in the biochemical analysis where it has been widely used to track the glucose levels in diabetic's patients. ANN is also capable of detecting pathological conditions such as tuberculosis. Image analysis is nowadays a core aspect of medicine. The need for proper image analysis cannot be over-emphasized.

Thus, ANNs have assisted in tumour detection and classification of chest X-rays. The results produced through the application of ANNs to medicine have been promising so far. Drug development, modeling, clinical research, pharmacoepidemiology, and medical data mining are some other medical areas where ANNs have been extensively used. It is important to mention that the high computation rates of ANNs had also contributed to its acceptance in medicine. Hence, paving way for ANN to be applied also in telemedicine. Having highlighted the importance of ANN in medicine, it is therefore important to ask if ANN can replace human experts? The answer is in negative- NO. ANN is thus a tool that is poised to help the doctors and researchers in the medical field. Finally, ANNs would assist the doctors in the screening process and ultimately to double-check and confirm their diagnosis.

3.2 Types of Neural Networks used in this research

3.2.1 Feed-Forward Neural Network (**feedforwardnet**)

Feedforward Neural Network, also known as *feedforwardnet* consists of layers. The network input layer connects the first layer. The output is produced through the last layer. The intermediate layers are connected in such a way that each intermediate layer has a connection from the previous layer. It is specifically used to map out the relationship between input-output. In the simplest form, the network has one hidden layer and numerous neurons. In terms of the syntax, *feedforwardnet* is given by:

Syntax

```
feedforwardnet(hiddenSizes,trainFcn)
```

Table 2. Feedforwardnet parameters.

hiddenSizes	Row vector of one or more hidden layer sizes (default = 10)
trainFcn	Training function (default = 'trainlm')

As pointed out, *feedforwardnet* maps input-output relationships. When more functionalities are required, specialized versions such as *fitnet* and pattern recognition are good choices.

Similarly, cascade feedfoward neural network (*cascadefeedforwardnet*) offers unique functionality as it connects the input layer to all other layers. Hence, fully established connections between layers are employed in *cascadefeedforwardnet*.

3.2.2 Elman Neural Network (elmannet)

The most widely cited example of feedforward network, also known as *feedforwardnet* is the Elman Neural Network (ENN) (Elman 1990). It was characterized by the fact that it has local memory and feedback connections. It was J.L Elman that first proposed it in 1990. It is also back propagation neural networks and a two-layer neural network. It basically consists of the input layer, hidden layer, and output layer respectively. The feedback connection is usually from the output of the hidden layer to its input as shown in **Figure 8**. It has a ready-made function called *elmannet* in MATLAB.

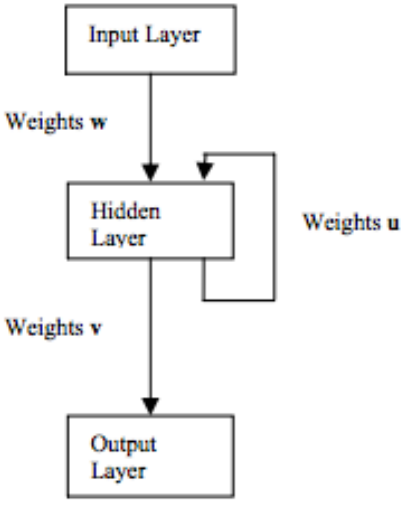


Figure 8. Block Diagram of the Elman Neural Network (Kannathal 2006).

The feedback connection ensures that Elman networks learn effectively. In addition, temporal and spatial patterns are easy to recognize and generate with the help of the feedback connection.

Mathematically, the algorithm of Elman neural network was presented in the equation below. However, it is important to mention that ENN uses *staticderiv* which was not a full dynamic derivative. The final output of the trained network is usually compared with the expected output to see how well and effective the network learned. Similarly, it is structurally represented as shown in **Figure 9**.

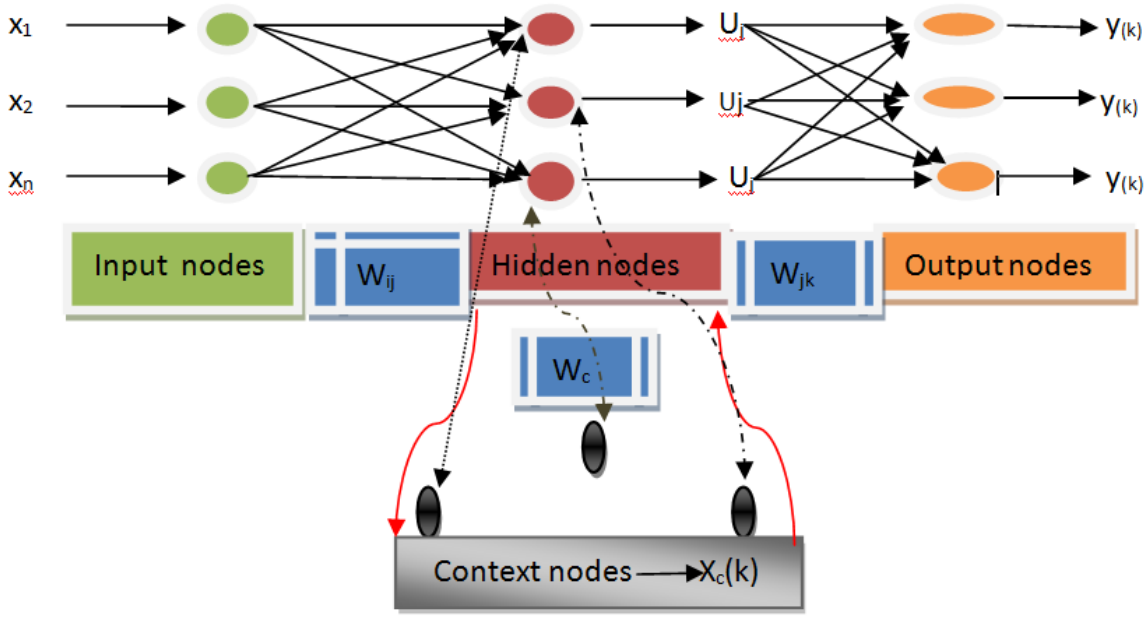


Figure 9. Structural representation of Elman neural network .

The structure of ENN was presented in **Figure 9**. It can be deduced from this figure that it has four nodes. The connections of input, hidden and output nodes are similar to the feed-forward network. As discussed in Chapter 2, the structure of Elman neural network consist of input nodes, hidden nodes, context nodes and output nodes respectively. The structure also includes the weights.

W_c represents the weight between context and hidden nodes.

W_{ij} denotes the weight between input and hidden nodes.

W_{jk} is the weight between hidden and output.

X_1.....Xn represents the inputs. Υ_j and Υ_k are activation functions (Hyperbolic tangents).

U_j is the hidden output. X_c (k) represent the context layer.

Θj and Θk biases in the hidden and output layers.

$Z_{(k)}$ represent the output

The feedback characteristic is a unique feature of Elman network and it basically utilizes the context node to memorize and return the hidden layer's output values. This essential feature makes Elman network to be sensitive and suitable for the learning purpose and also in the analysis of time series data and historical data respectively.

In terms of the syntax, Elman network is given as follows:

Syntax

```
elmannet(layerdelays,hiddenSizes,trainFcn)
```

Table 3. Syntax parameters of Elman network.

layerdelays	Row vector of increasing 0 or positive delays (default = 1:2)
hiddenSizes	Row vector of one or more hidden layer sizes (default = 10)
trainFcn	Training function (default = 'trainlm')

Based on the syntax and the structural representation of Elman neural network, the output of each nodes can be mathematically modelled as follows:

$$X_c(k) = \alpha Xc\ (k-1) + U(j-1) \quad (1)$$

$$U_j = f\ [\ (W_c * X_c(k)) + (Wij * Xn)\ ] \quad (2)$$

$$y_{(k)} = g\ [(W_{jk} * U_j)] \quad (3)$$

Equations I-3 above represents the outputs of context, hidden and the final outputs respectively.

Additionally, f() and g() in the equation above are the linear and nonlinear output functions of the output nodes and hidden nodes respectively. Equations (I-3) can be modified further in such a way that when the input vectors are mapped to set of hidden nodes through an activation function Γ_j, the mathematical representation is as shown in equation 4.

$$u_j = \Gamma_j\left[\sum_{i=1}^{M} w_{ij}u_j + \theta_j + u_j'\right]. \qquad (4)$$

Since Elman has context layer, it means that there will be a delayed hidden variables represented as U_j' from the prior training iteration. Thus, the result of the mathematical modification is as shown above. In the same way, the output layer can be modified further through similar procedure as explained for the hidden layer. In this case, the activation function is given as Γ_k. Finally, the mapping relationship between hidden and output layer is represented as y_k and given in equation 5 as :

$$y_k = \Gamma_k\left[\sum_{i=1}^{N} w_{jk}u_j + \theta_k\right]. \qquad (5)$$

With the introduction of full dynamic derivative calculations that uses the concept of *fpderiv and bttderiv,* time delay neural networks (timedelaynet), layer recurrent neural network (layrecnet), nonlinear autoregressive neural (NARXNET) and nonlinear autoregressive neural network with external inputs (NARXNET) are now the preferred networks because they produced better error performance than Elman neural network. These networks would be examined in Chapter 4.

3.2.3 Time delay neural network (*timedelaynet*)

Time delay neural network, TDNN was first developed in the 1980s (Waibel et al 1989). It is an artificial neural network that is characterized by two special layers. These are the hidden layer and output layer. In TDNN, the nodes are connected fully by direct connections as shown in **Figure 10**.

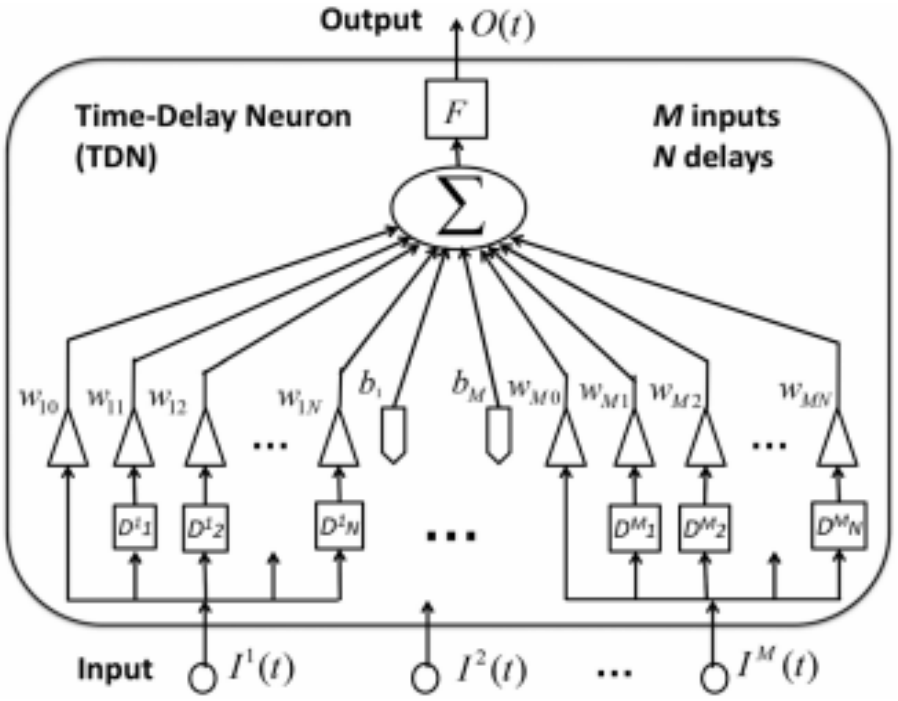


Figure 10. Single time delay neuron (TDN) with inputs and delays at the time (t) (Hongying et al 2016).

The nodes of hidden layer and output layer are time-delay neurons (TDNs). The inputs are multiple inputs, time series with time step (t). The inputs are grouped as M inputs but explicitly it is made up of $I^1(t)$ until I^M (t). Each of the explicit inputs has a bias value represented as bi.

Similarly, as shown in Figure 10, TDNN also has N delays explicitly as D^i_1 till D^i_N. Since it is the time delay, it is capable of storing previous inputs $I^i(t-d)$ where d varies from 1 to N. Also, N is the independent unknown weights represented as $w^i_1 ... w^i_N$. F is the transfer function of f(x). The output O(t) is represented by the equation below:

$$O(t) = f\left(\sum_{i=1}^{M}\left[\sum_{d=0}^{N} I^i(t-d) * w_{id} + b_i\right]\right).$$

From the output equation above, the overall outcome of the neurons could be considered to be dependent on the current time (t) and also the previous time steps (t-d).

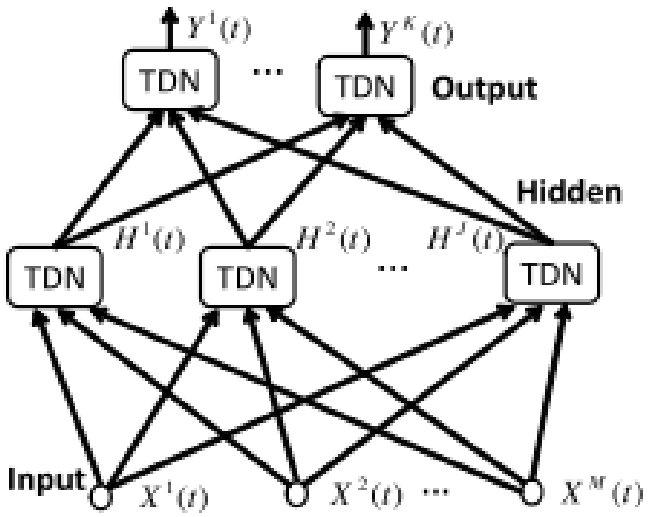


Figure 11. The Architecture of TDNN neural network (Hongying et al 2016).

Where:

W_{id}^j weight of the hidden node $\mathbf{H^j}$.

W_{jd}^r weight of the output node $\mathbf{O^r}$

b_i^j and c_i^r are biases.

N_1 number of delays for output layer.

N_2 number of delays for hidden layer.

Having understood the single TDN it becomes easier to model dynamic nonlinear characteristics of series inputs. This forms the basic building block of TDNN. In terms of architecture, TDNN has hidden layer with J TDNs and conversely an output layer with R number of TDNs that are fully connected as shown in **Figure 11**.

$$O^r(t) = f\left(\sum_{j=1}^{J}\left[\sum_{d=0}^{N_1} H^j(t-d) * v_{jd}^r + c_j^r\right]\right), r = 1, 2, \ldots, R$$

$$H^j(t) = f\left(\sum_{i=1}^{M}\left[\sum_{d=0}^{N_2} X^i(t-d) * w_{id}^j + b_i^j\right]\right), j = 1, 2, \ldots, J$$

TDNN can be trained using Levenberg-Marquardt algorithm (Levenberg 1944; Marquardt 1963). Levenberg-Marquardt is an example of the traditional feedforward-feedback network. The training process of Levenberg-Marquardt optimizes the weights through iterations. Input time series (X) and known labels Y(t) are iterated for t = 1,....T, given that T is the length of the sequence.

3.3 Layer Recurrent Neural Network (layrecnet)

Layer recurrent neural networks (LRNN) are dynamic and artificial neural network. In this network, there are connections between units, and these connections are in a directed cycle manner. Hence, the name - dynamic neural network. LRNN is similar to feedforward network but differs in the sense that each layer has a recurrent connection with a tap delay associated with it. This is an important feature that makes LRNN to have an infinite dynamic response to time series input data.

However, when a finite input responses are desired, time delay (timedelaynet) and distributed delay (distdelaynet) neural networks are the neural networks of choice.
In terms of the syntax of LRNN, it is given by:

Syntax

```
layrecnet(layerDelays,hiddenSizes,trainFcn)
```

Where:

Layrecnet (layer recurrent network) takes the following arguments as shown in **Table 4.**

Table 4. Layrecnet parameters

layerDelays	**Row vector of increasing 0 or positive delays (default = 1:2)**
hiddenSizes	Row vector of one or more hidden layer sizes (default = 10)
trainFcn	Training function (default = 'trainlm')

With the above parameters, the layrecnet function returns a layer recurrent neural network. Though ELMAN is a simplified form of LRNN, LRNN is characterized by the fact that there is a feedback loop, with a single delay, around each layer of the network except for the last layer as shown in **Figure 12**.

There are different types of LRNN such as fully recurrent network, recursive neural networks, Hopfield Network, Elman and Jordan Neural Networks, continuous-time RNN, Bi-directional RNN to mention but a few.

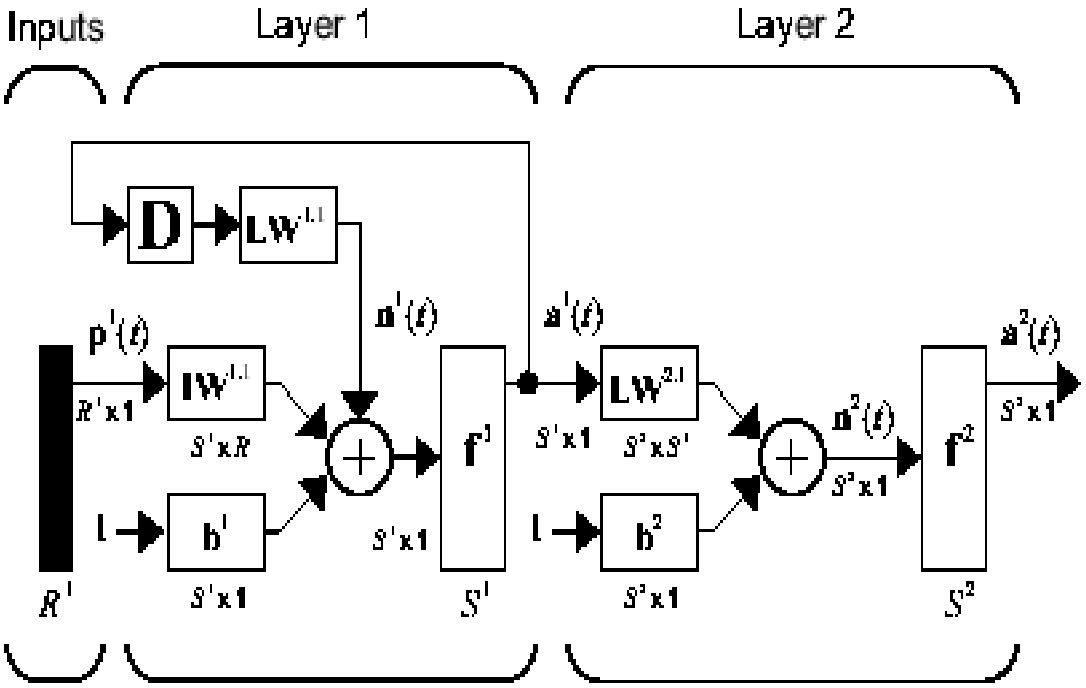


Figure 12. Layer recurrent network architecture (MATLAB 2017).

3.3.1 Fully Recurrent Neural Network

The Fully recurrent network is a network of neuron-like units developed in the 1980s. Each of the units has a directed connection to every other unit. Each connection has a real-valued weight that is modifiable. Similarly, each unit is characterised by a time varying real-valued activation.

3.3.2 Hopfield Neural Network (HNN)

It is a form of recurrent artificial neural network invented in 1982. It was named after the inventor, John Hopfield. It has an essential feature that it guarantees that all its dynamics will converge, that is, to the local minimum. Albeit, it sometimes converges to false local minimum. It is not used for sequence of patterns and this makes it to be a unique neural network. All connections are symmetry therefore, it is designed specifically to require stationary inputs as shown in **Figure 13**.

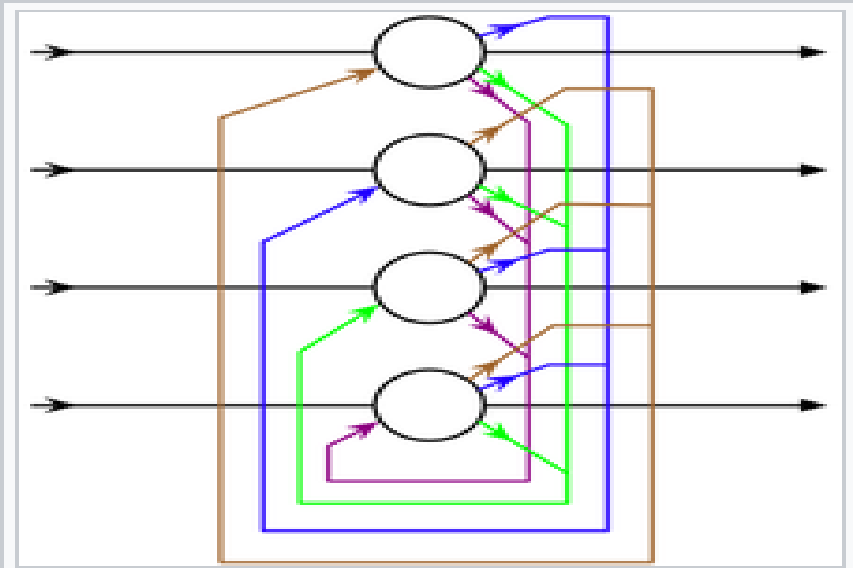


Figure 13. A four nodes Hopfield neural network (Hopfield 1982)

Hence, connections in Hopfield network have two restrictions. These are (i) no unit has a connection with itself (ii) connections must be symmetry. It has few variations such as bidirectional associative memory (BAM). Hopfield neural network basically takes two values for their states, that is, 1 and -1. These values are determined by whether the inputs exceed the threshold or not. If the inputs exceed the threshold, it is +1, but if it is within the threshold, it is -1. However, 0 and 1 values are used in some literature.

Updating one unit in the HNN is based on the under listed rules:

$$s_i \leftarrow \begin{cases} +1 & \text{if } \sum_j w_{ij} s_j \geq \theta_i, \\ -1 & \text{otherwise.} \end{cases}$$

Given that:

w_{ij} is the weight of the connection from unit j to unit i.

S_j state of unit j.

Θ_i is the threshold of unit i.

HNN units can be updated either through synchronous and asynchronous means. By synchronous, it means that all the units are updated at a time. This method has a disadvantage that it is not effective. Similarly, in asynchronous, only one unit is updated at a time. The unit to be updated can be randomly picked or in a pre-defined order. With regards to the learning rules in HNN, it can either be local or incremental learning rules respectively. Learning or training rule is considered local in HNN if each weight is updated based on the information available to neurons on either side of the connection associated with a particular weight. Conversely, in incremental learning rules, the new pattern can be learned without using information from the previous patterns.

3.3.3 Recursive Neural Network

As the name implies, same sets of weights are recursively applied over a structure. The architecture of a simple recursive neural network is as shown in Figure **14**.

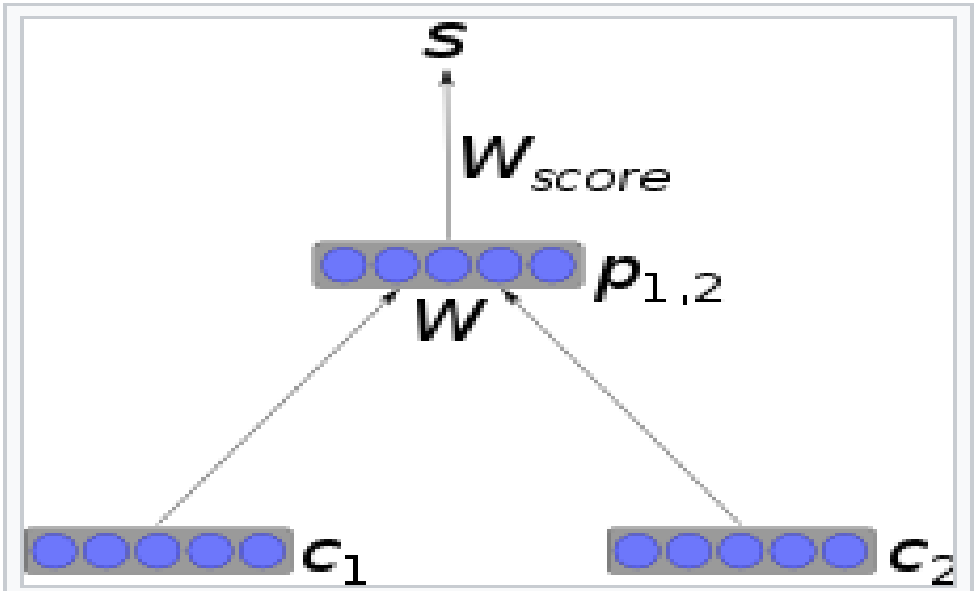


Figure 14. An architecture of recursive neural network (Hammer et al 2004).

The RNN was trained using one of the widely used algorithms like scaled conjugate gradient. The gradient is calculated using backpropagation through structure (BPTS) which is a family of backpropagation through time (BPTT) used in recurrent neural networks (Goller and Kuchler 1996). In addition, recurrent and recursive neural network differs. In recurrent neural network, the hidden representation and previous time step are combined to produce a unique representation of the current time step. Also, the chain of the recurrent neural network is linear. Recursive neural network, on the other hand, operates on the hierarchical structure where parents representation arises when the child representations are combined (Hammer et al 2004).

3.4 Nonlinear autoregressive neural network (NARNET)

NARNET is used to make predictions of a time series based on that particular series past values (Nyanteh et al 2013 and Lopez 2012). In MATLAB, nonlinear autoregressive neural network has a function known as *narnet*. It takes the following arguments:

narnet(feedbackDelays,hiddenSizes,trainFcn)

Table 5. Parameters for narnet

feedbackDelays	Row vector of increasing 0 or positive delays (default = 1:2)
hiddenSizes	Row vector of one or more hidden layer sizes (default = 10)
trainFcn	Training function (default = 'trainlm')

Modelling time series using linear model is more often than not a difficult task. This is because time series applications have high variations and fast transient durations. Therefore, the need for a better model for such applications becomes imperative. NARNET has been touted to handle some of the complications that are peculiar to time series data. NARNET is given by:

$$y(t) = h(y(t-1),\ y(t-2),\ \ldots,\ y(t-p)) + \epsilon(t)$$

The equation above explains how predictions could be made using NARNET. For example, predictions can be made using past values of the data series (Ibrahim et al 2016). Where y(t) is the predicted value, the function known as ***h (previous values)*** is not known but the past values of y(t) can make the prediction to be realizable.

In the world of neural network, with network training, the function h() can be achieved by means of the weights and bias optimizations respectively. Where ε(t) denotes the error of the approximation of the series. The structure of the NARNET is as shown in **Figure 15**.

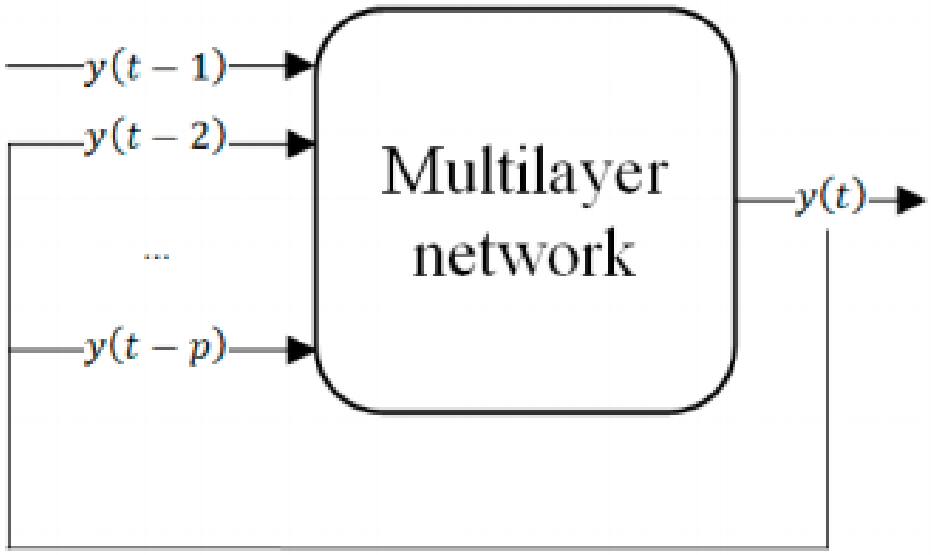


Figure 15. Nonlinear autoregressive network (NARNET) (Luiz Gonzaga et al 2016)

The past values, that is, the *p* values ***y(t-1).....y(t-p)*** are known as the feedback delays. To obtain the network topology that can provide the best performance, several factors need to be considered. Example of such factors include: the training algorithm, number of hidden layers and neurons to mention but a few. Usually, adjusting the number of hidden neurons or changing the training algorithm normally gives better performance. The number of hidden layers and training algorithm are adjusted and varied through trial-and-error means. It is nevertheless important to understand that the complexity of the system has a direct proportionality to the number of neurons. As increasing the number of neurons increases, the network also becomes more complex.

Although, an increased number of neurons gives better generalization efficiency and the speed of computation of the network. NARNET is based on Levenberg-Marquardt propagation procedure (LMBP) algorithm (Alwakeel & Shaaban 2010; Marquardt 1963; Hagan et al 1996).

It is the default because it is the fastest type of backpropagation algorithm. In addition, the training duration is very fast and it uses the second-order derivative, hence, there is no need to compute Hessian Matrix. Instead, it uses the Jacobian Matrix for calculation. NARNET uses either Mean Square Error (MSE) or Error Sum of Squares (SSE) and both are given by:

$$SSE = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

$$MSE = \frac{SSE}{n}$$

Where:

y_i is i-th data sample.
ý approximated data obtained by the network for the value y_i.
n is the number of the data sample.

In this thesis work, layer recurrent neural network would be used to predict of recurrence and mortality of tongue cancer in the patients. Both NARNET and NARXNET can be considered as a recommendation for future work.

3.5 Nonlinear Autoregressive Neural Network with external input (narxnet)

This is similar to NARNET but differs with the addition of external input. It is also a nonlinear model that predicts the future values based on the past values and also an exogenous or external data supplied to the network. In terms of syntax, it is given by:

Syntax

```
narxnet(inputDelays,feedbackDelays,hiddenSizes,trainFcn)
```

Where:

Table 6. NARXNET parameters

InputDelays	**Row vector of increasing 0 or positive delays (default = 1:2)**
feedbackDelays	Row vector of increasing 0 or positive delays (default = 1:2)
hiddenSizes	Row vector of one or more hidden layer sizes (default= 10)
trainFcn	Training function (default = 'trainlm')

In NARX network, known as NARXNET, the network is able to predict series y(t) given the past values of series y and another external series x(t) as shown in the equation below:

$$y(t) = h(x(t-1),\ x(t-2),\ \ldots,\ x(t-k),\ y(t-1),\ y(t-2),\ \ldots,\ y(t-p)) + \epsilon(t)$$

The external or exogenous series could be single or multidimensional. The architecture for NARXNET is as shown in Figure

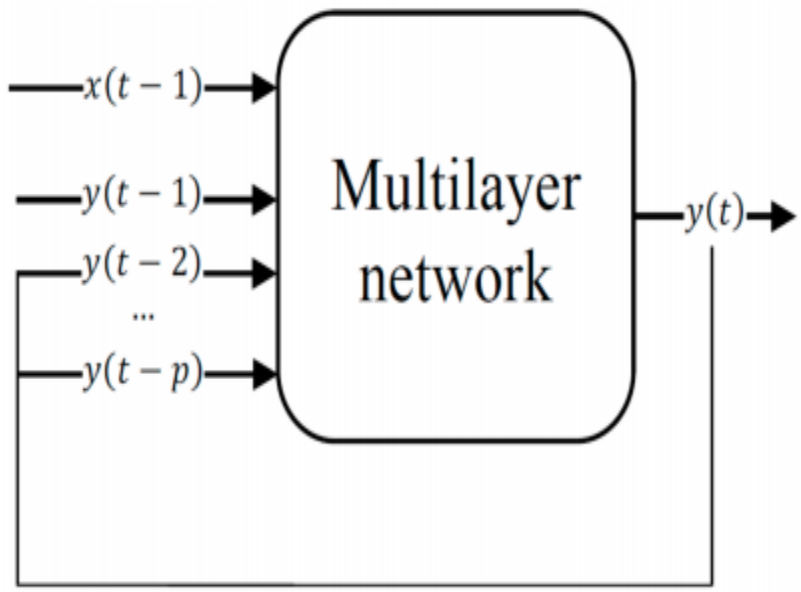


Figure 16. The architecture of nonlinear regressive network with external inputs (Luiz Gonzaga et al 2016).

NARXNET AND NARNET are quite similar but differs only with the addition of external inputs in the case of NARXNET. The training of NARXNET equally uses LMBP. NARXNET produces better performance than NARNET but the complexity nature of NARXNET has made NARNET to be widely preferred (Safavieh et al 2007). Therefore, in this thesis work, both methods will be used in the analysis of the dataset.

4 ANN SIMULATION OF FIXED AND DYNAMIC DATASETS

This chapter is aimed at examining the performance analysis, error estimation and most importantly to have a clear understanding of some of the neural network types discussed in **Chapter 3**. This is in preparation for the neural networks for the real data, that is, the case study to be examined in **Chapter 5** of this thesis. The dataset used in this chapter was provided by my supervisor, Professor Elmusrati. The dataset has been generated by a certain algorithm known to him. My main task is to test the dataset with the enumerated neural network types discussed in **Chapter 3**.

The datasets were sent in two batches. The first batch of dataset was fixed dataset, while the second batch was dynamic dataset. The datasets were sent in excel formats (xlxs) and the datasets can be found as Appendixes I and II respectively in the appendix section of this thesis write up. By fixed dataset, it means that the values within the rows and columns of the dataset are related by a direct equation without the need for any past or future values. Conversely, dynamic datasets were obtained by establishing a relationship with either the past or future value or both as the case may be. In both cases, especially in the dynamic datasets, it is possible that the values of the columns and rows change each time that the files/documents are opened. Thereby giving false results. To avoid this error, the file was opened once and it was imported directly into MATLAB workspace.

Therefore, **Chapter 4** is poised to look at the neural network with regards to dynamic datasets only. Finally, in this chapter, the MATLAB code, various plots of performance analysis, regression plots, and expected and trained values plots will be shown without interpreting the plots. This is because the main work of this thesis is in **Chapter 5** and that is the main dataset of interest. All the analysis and explanations of each plot will be explained in **Chapter 5**.

4.1 Simulation exercise with Feedforward Neural Network

Using the dynamic dataset as contained in Appendix II, the following commands were issued to MATLAB as shown in **Figure 17**.

Command Window

```
Unable to delete file: C:\Users\Alabi\Desktop\muadh_dydata.xls
>>
>> A = (dinputs)';
>> B = (douts)';
>>
>> net = feedforwardnet(10);
net = train(net,A,B);
view(net)
y = net(A);
perf = perform(net,y,B)

perf =

   1.0618e-09
```

Figure 17. Feedforwardnet MATLAB code window.

From **Figure 17**, the performance error was very small. The learning was thus successful.

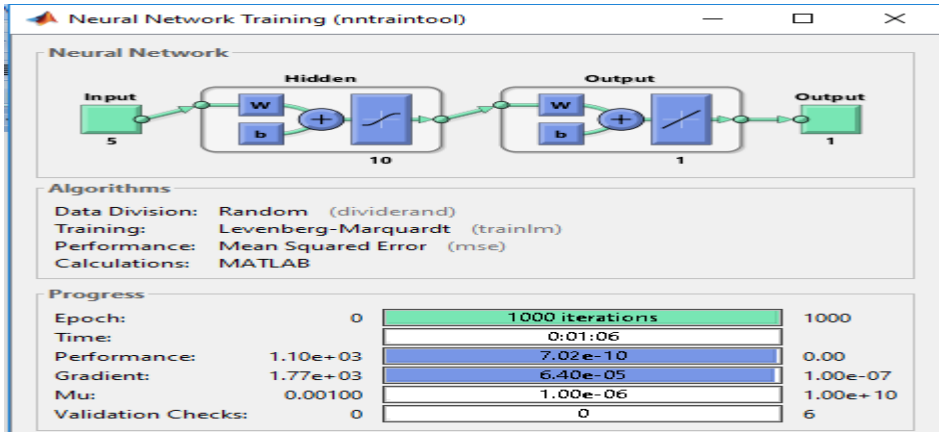


Figure 18. Neural network training output

It is worthy of note that some of the code might be missing from **Figure 17**. The output shown in **Figure 17** is a truncated output.

Similarly, **Figure 18** showed the neural network training output.

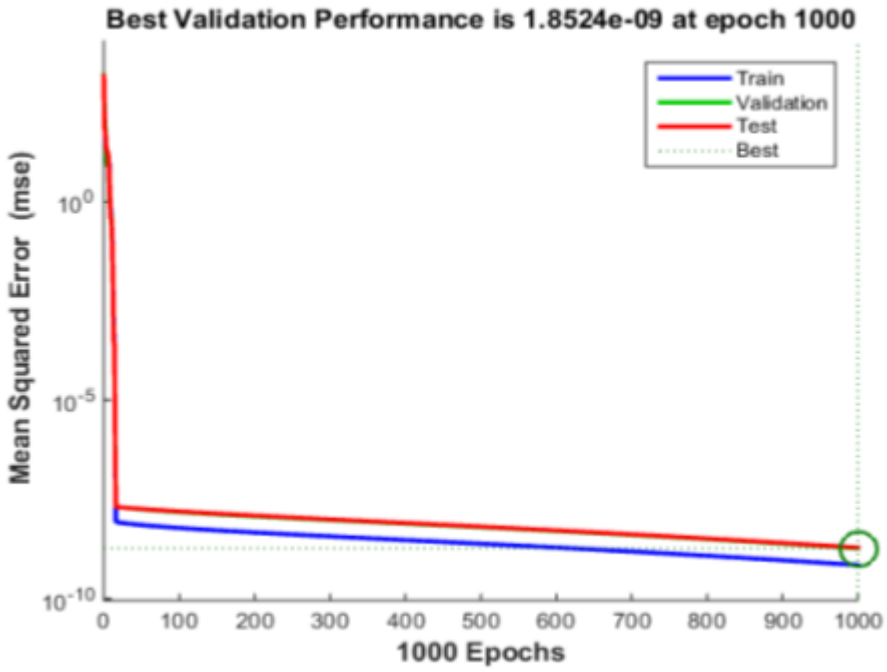


Figure 19. Performance error plot for feedforwardnet.

The network was trained using Levenberg-Marquardt algorithm (trainlm) and the error from the training was calculated using Mean Squared Error.

```
Bold =

    targetOutputs    NeuralOutputs
    _____________    _____________

     -0.80141         -0.80141
      -5.0418          -5.0418
       3.4762           3.4762
      -21.754          -21.754
       3.8645           3.8645
      -6.9113          -6.9113
      -10.667          -10.667
       4.7497           4.7497
      -15.963          -15.963
      -11.268          -11.268
       3.0855           3.0855
      -1.9639          -1.9639
      -15.908          -15.908
      -5.6676          -5.6677
```

Figure 20. Target outputs and the neural outputs

The dataset was divided randomly using *dividerand* output. Hence, no need to manually divide the data into training, validation, and testing. However, the network can be trained manually using the *tr.trainInd*, *tr.valInd* and *tr.testInd*. These functions can be used to divide the data into training, validation and testing respectively. The plot from the training is as shown in **Figure 19**.

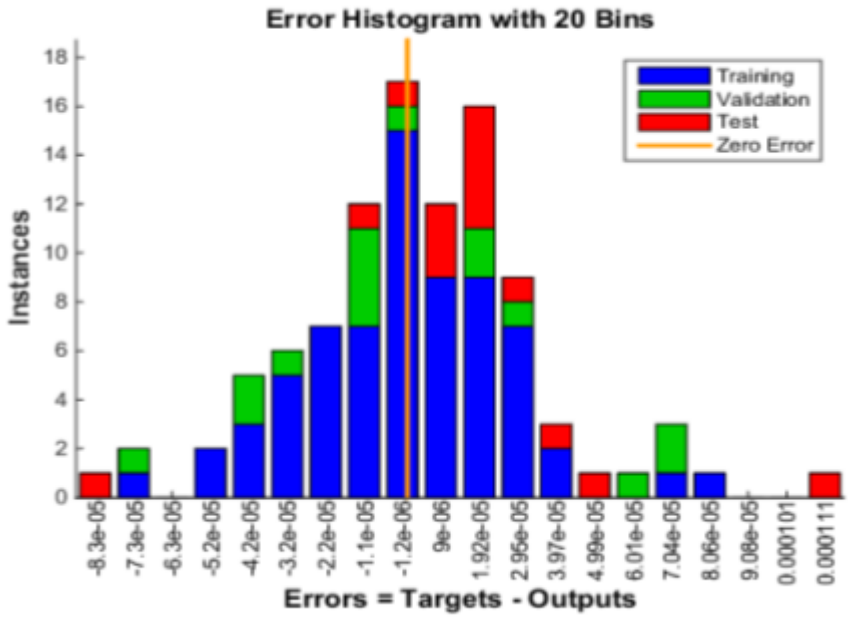


Figure 21. Error histogram of the targets and the neural outputs.

The variation between the target output and the neural outputs is as shown in **Figure 20**. Following **Figure 20**, the difference between the target and the neural output gives the error as shown in error histogram of **Figure 21**. The network training state is shown in **Figure 22**. It is a plot that shows the gradient, validation check, and epoch. The epoch gives the iteration level at which the network validation performance reached the minimum. In this case, as shown in **Figure 22**, the network validation reached the minimum at about 1000 epoch.

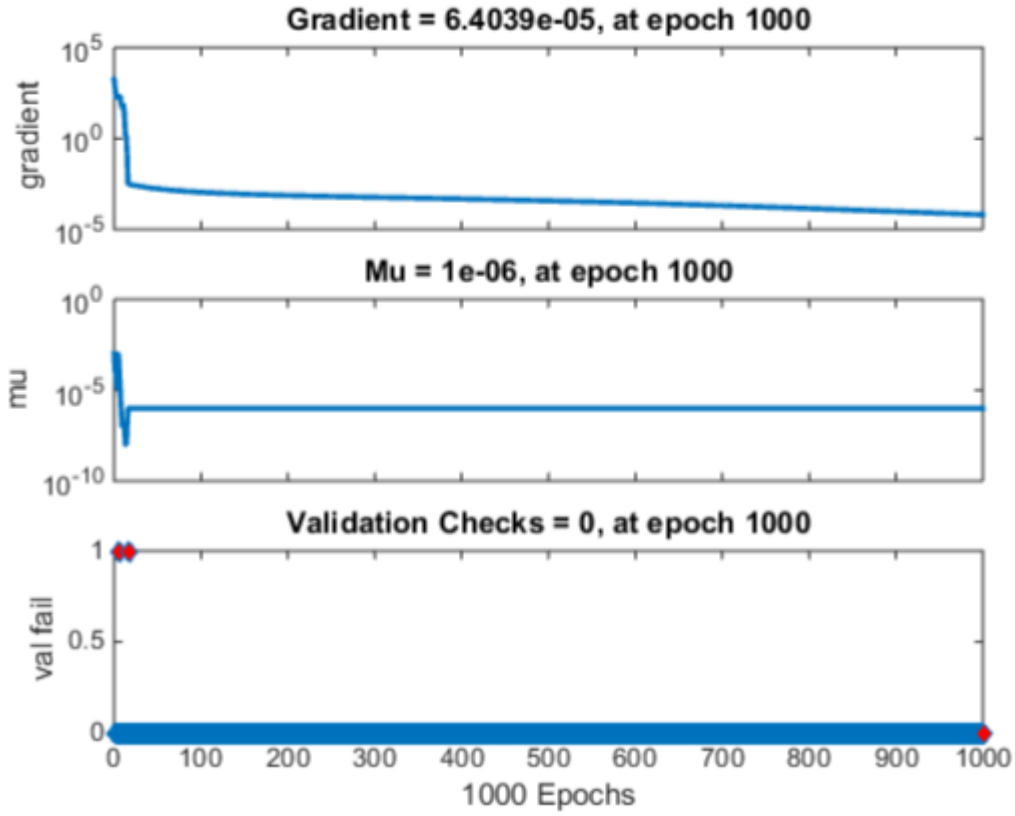


Figure 22. Training state of the network.

It is important to mention that it is possible to have errors after the training and the results might not be as expected. To improve the results in this case, it is always a good practice, to initialize the network again and the training can be performed again.

This is because, each time the network is initialized especially a *feedforwardnet*, the network parameters are different and thus the results might be different on each occasion. Also, the number of hidden neurons can be increased. It is important to mention that the number of hidden layer neurons should not be unnecessarily large to avoid under-characterization issues. Finally, the training algorithm can be changed may be from Levenberg-Marquardt to Bayesian regularization training.

Regression plot is shown in **Figure 23**. It shows the relationship between the variables that made up of the inputs and most importantly, how the inputs and outputs are related. The dashed line in the plot indicates how the expected output (target output) relates with the trained outputs (neural outputs).

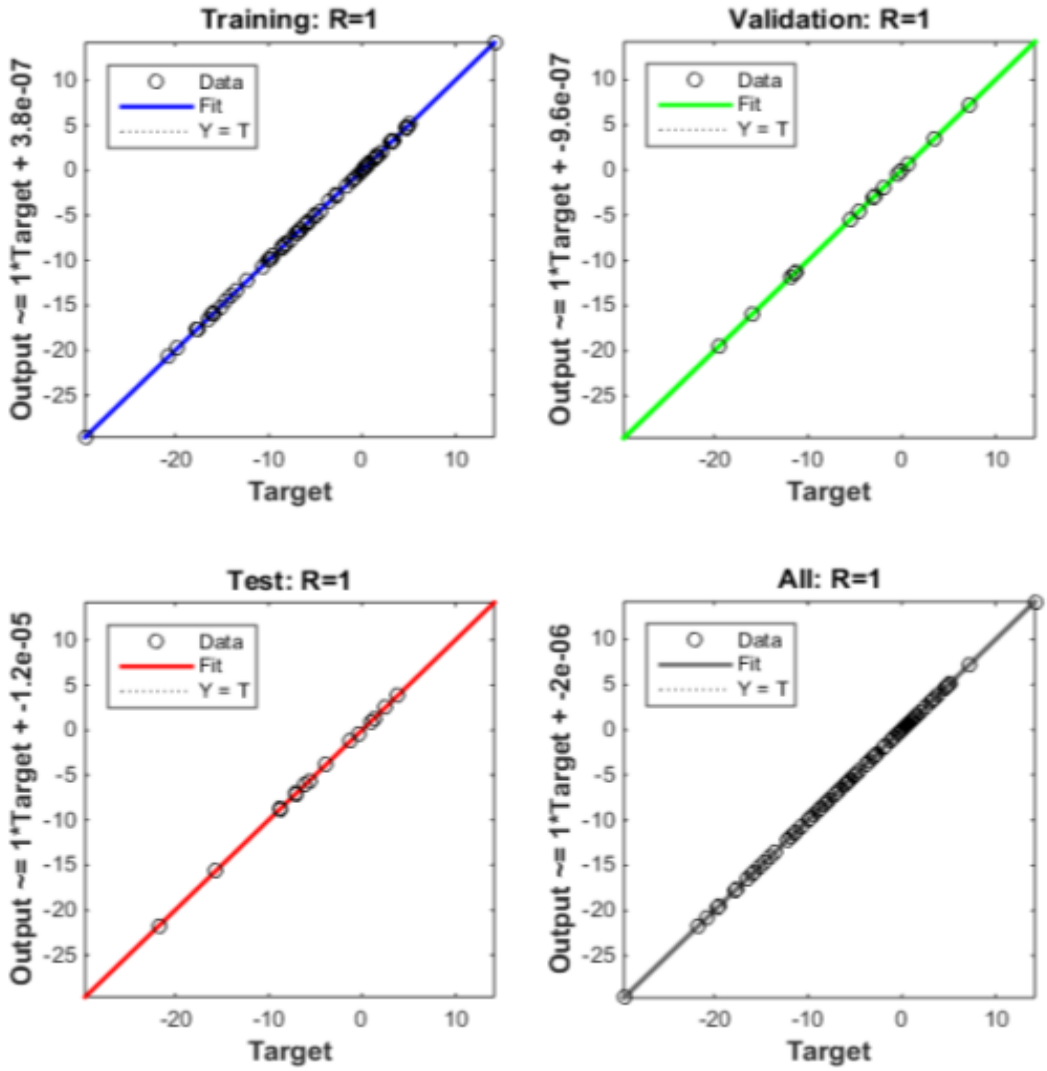


Figure 23. Regression analysis of the network training.

While the solid line indicated the linear regression between outputs and targets. The regression value, R, as shown in Figure **23** for the training, testing and validation had the value of 1.

Hence, it is an indication of an exact linear relationship between the targets and the neural outputs. Finally, a simple plot of the expected or target output is shown against the neural output in **Figure 24**. It can be seen from **Figure 24** that there is no much difference between the target output and the expected output plots. It is quite difficult to observe any difference in the plots.

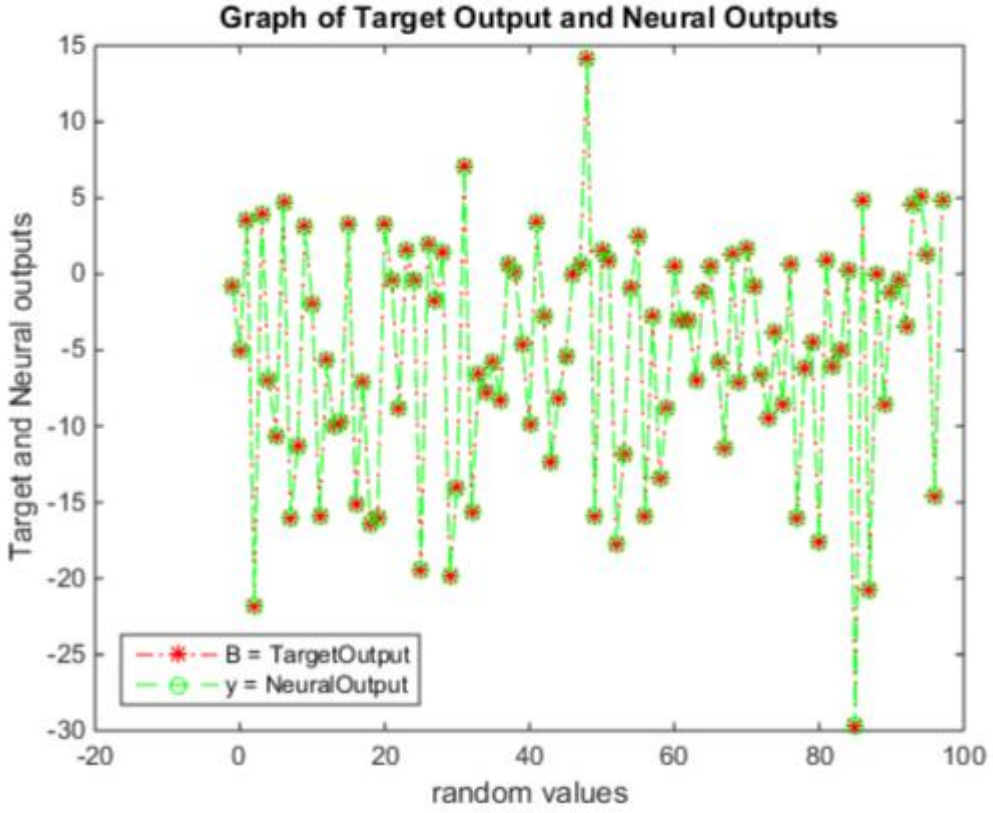


Figure 24. Plot of Target and Neural Outputs of feedforward network

This is because the network effectively learned the relationship between the inputs and the output. Consequently, the performance error was quite insignificant as shown in **Figure 17**. Similarly, the regression plot of **Figure 23** was also a strong indication to the fact that the network had effectively learned the relationship between the input variables and the output.

4.2 Simulation using Elman Neural Network

From the dataset contained in Appendix II, the network performance of the Elman neural network can be examined. Unlike the feedforward neural network that uses the function *feedforwardnet*, Elman neural network uses the *elmannet* function with the performance evaluation done by another function called *preparets.* Though Elman neural network is also a variant of *feedforwardnet*, the difference lies in the training function. The dataset was randomly divided using *dividerand* but the default training function for *elmannet* was Gradient Descent With Momentum and Adaptive LR using the *traingdx* function.

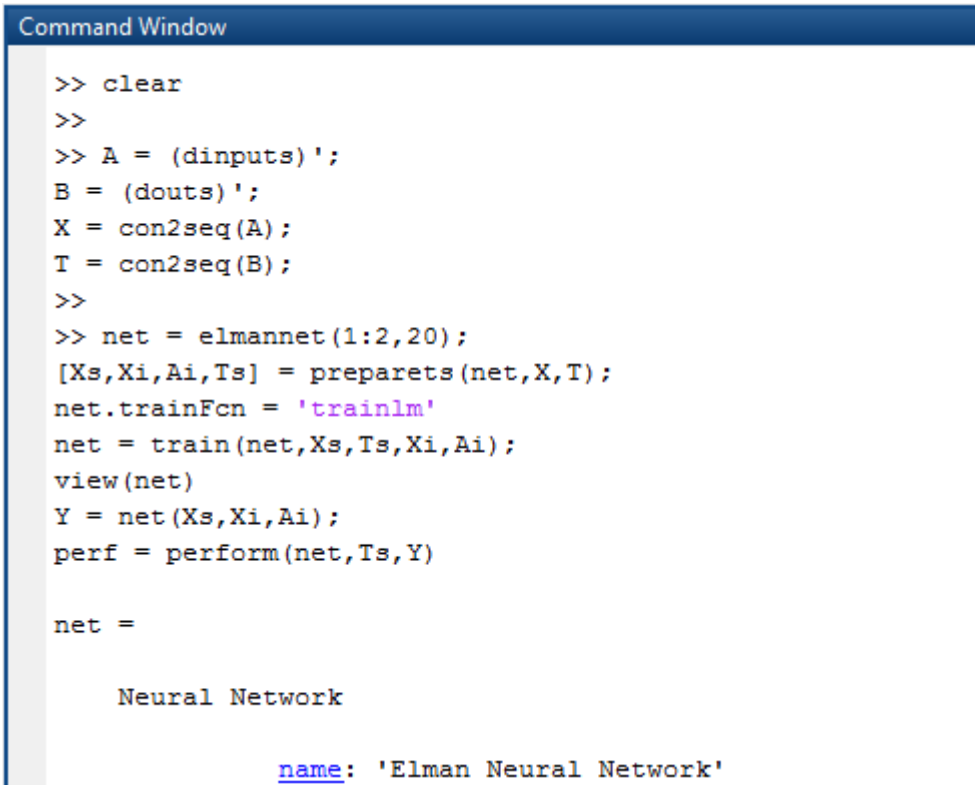


Figure 25. Elman neural network command window

MATLAB Command window of Elman neural network is as shown in **Figure 25**. It was observed that this function did not train the network effectively, thus, I changed the training function to Levenberg-Marquardt function of *trainlm*.

Furthermore, the number of hidden neurons was increased from 10 to 20 to ensure a better performance as shown in **Figure 26**.

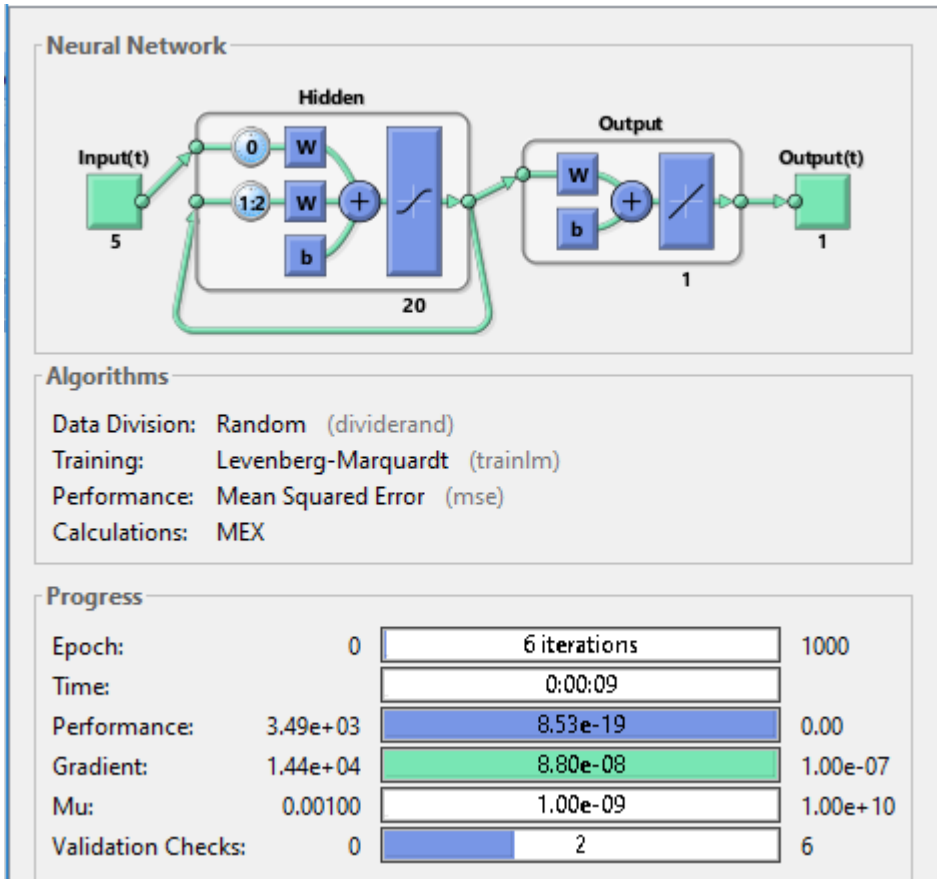


Figure 26. Training window for Elman neural network.

The truncated outputs of the target and the trained output, that is , neural output is shown in **Figure 27**. Despite the change in the training algorithm and also increasing the number of hidden neurons, Elman neural network did not learn the relationship between the inputs and the outputs effectively. This is because Elman neural network uses simplified derivatives calculations known as *staticderiv*.

This simplified derivative calculations used by Elman is known to ignore delayed connections. Thus, the learning was not as efficient as the feedforwardnet discussed in **Section 4.1**.

Bold =

targetOutputs	NeuralOutputs
[3.4762]	[3.4757]
[-21.7538]	[-21.7554]
[3.8645]	[8.5896]
[-6.9113]	[-6.9117]
[-10.6668]	[-10.6636]
[4.7497]	[25.2816]
[-15.9633]	[-15.9711]
[-11.2684]	[-2.7756]
[3.0855]	[3.0906]
[-1.9639]	[-1.9580]
[-15.9083]	[-15.9048]
[-5.6676]	[-3.0988]
[-10.0151]	[-10.0279]
[-9.7302]	[-9.7677]
[3.1961]	[3.1583]
[-15.1480]	[-15.1483]
[-7.0526]	[-7.0691]
[-16.4780]	[-16.4880]

Figure 27. Target and Neural outputs of the Elman neural network.

Therefore, the need to check other network becomes imperative. The advent of full derivative calculations such as *fpderiv* and *bttderiv*, gives the researchers wide range of neural network to choose from to enhance better performance. Based on the failure of Elman neural network to performed the learning effectively, it is important to examine another network such as *timedelay, layrecnet, narnet, narxnet* for better training and efficient prediction of inputs-outputs relationships. These will be shown in subsequent Sections within Chapter **4** of this thesis. To probe further on why Elman did not learn as expected, the network performance of the network is shown in **Figure 28**. This plot shows the performance of the data when divided into training, validation and testing.

Mean Squared Error was used in the calculation of the training performance and as shown in **Figure 28**; it was at 6 epoch iterations where the best fit occurred at 4th iterations.

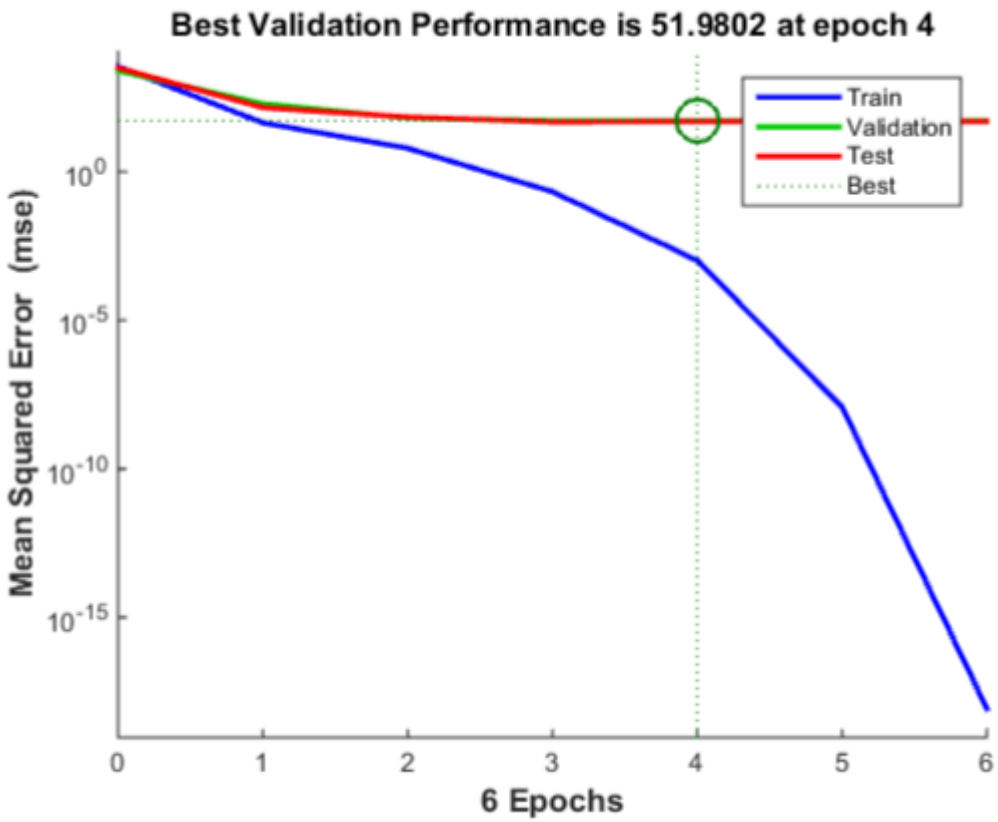


Figure 28. Training performance of Elman neural network.

The regression plot of **Figure 29** clearly showed that there exists some level of relationships between the training, testing and validation data respectively. The regression value, R was not exactly 1 but closer to 1, which means that there is a relationship between the inputs and the output to a large extent.

In furtherance, the 'all plot' that was shown on the regression plot in **Figure 29** is an indication of the extent of the relationship between the inputs and output.

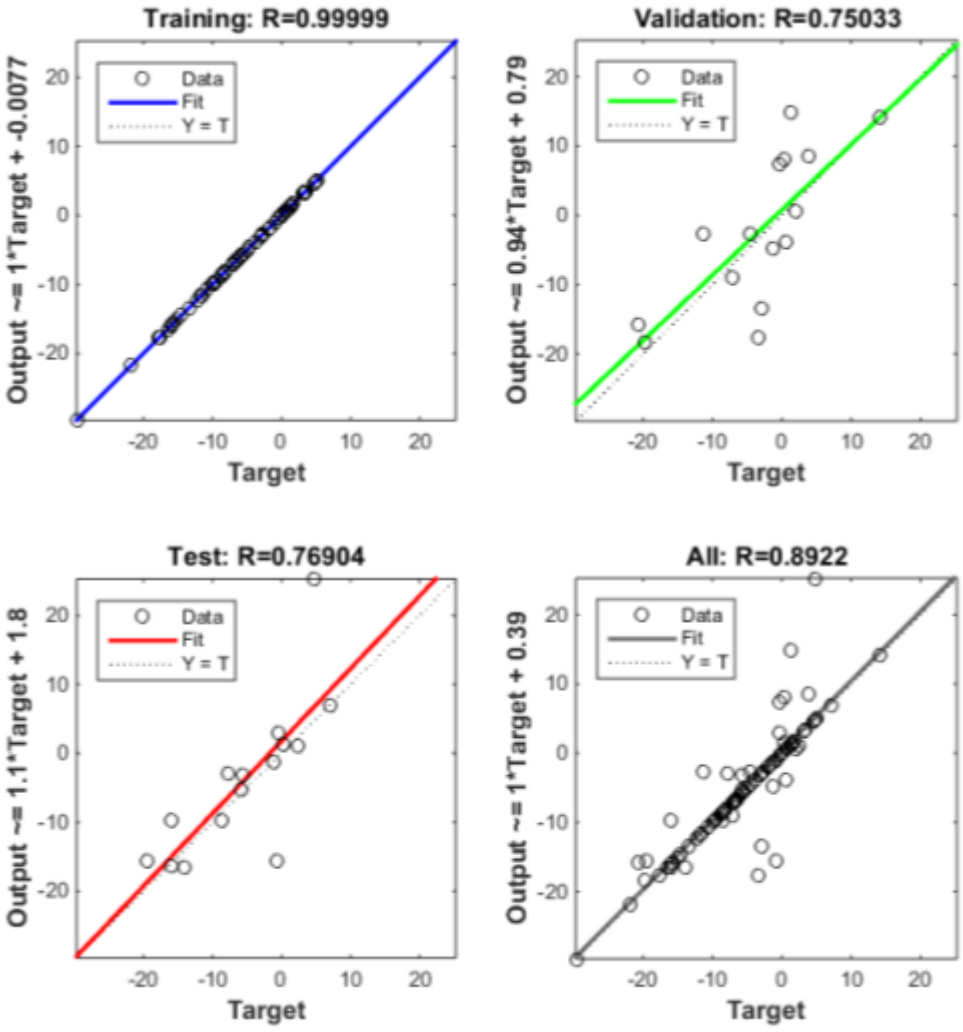


Figure 29. Regression plot of Elman neural network training

The cluster points on each point showed how many of the variables that have no relationship between the inputs and the outputs. Interestingly, the training was thorough as shown on the training plot of the regression plot.

Finally, to put the difference between the target and the neural outputs succinctly, **Figure 30** provides a quick view of the variation in the target and the neural output.

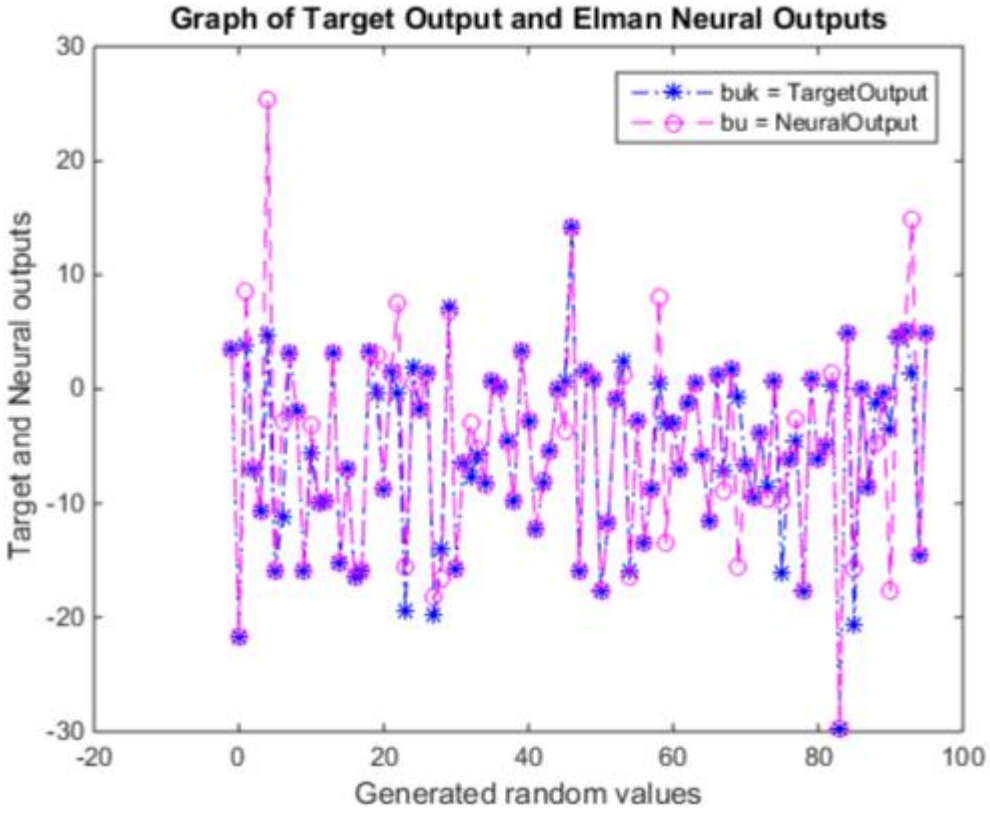


Figure 30. Elman neural network plot of target and neural outputs

Based on Figure 30, it can be seen that variation occurs only on few occasions. That is, about 8 times (8 rows) out of the 100 rows considered. Elman neural network learned well but it was not as effective as with the case of feed-forward neural network.

4.3 Timedelay Neural Network's simulation exercise

Timedelay neural network is of paramount importance from the point of view that the input weights have a tap delay line that is associated with it. Hence, it is the mostly preferred in time series input data because it has a finite dynamic response. Levenberg-Maquardt was the training method used and the dataset was randomly divided using *dividerand*. The command window implementation of timedelay neural network is shown in **Figure 31.**

```
    [-14.5608]        [ -7.8275]
    [  4.8801]        [   7.2224]

>> clear
>>
>> A = (dinputs)';
B = (douts)';
X = con2seq(A);
T = con2seq(B);
>>
>> net = timedelaynet(1:2,10);
[Xs,Xi,Ai,Ts] = preparets(net,X,T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
Y = net(Xs,Xi,Ai);
perf = perform(net,Ts,Y)

perf =

   38.7236

fx >>
```

Figure 31. Command window for time delay neural network.

Both the training algorithm and the number of hidden neurons were changed to ensure that the network effectively learned the relationship. The training algorithm used is as shown in **Figure 32**.

However, these efforts of changing the training algorithm and increasing the number of hidden neurons have no positive impact on the network as far as improving the performance was concerned.

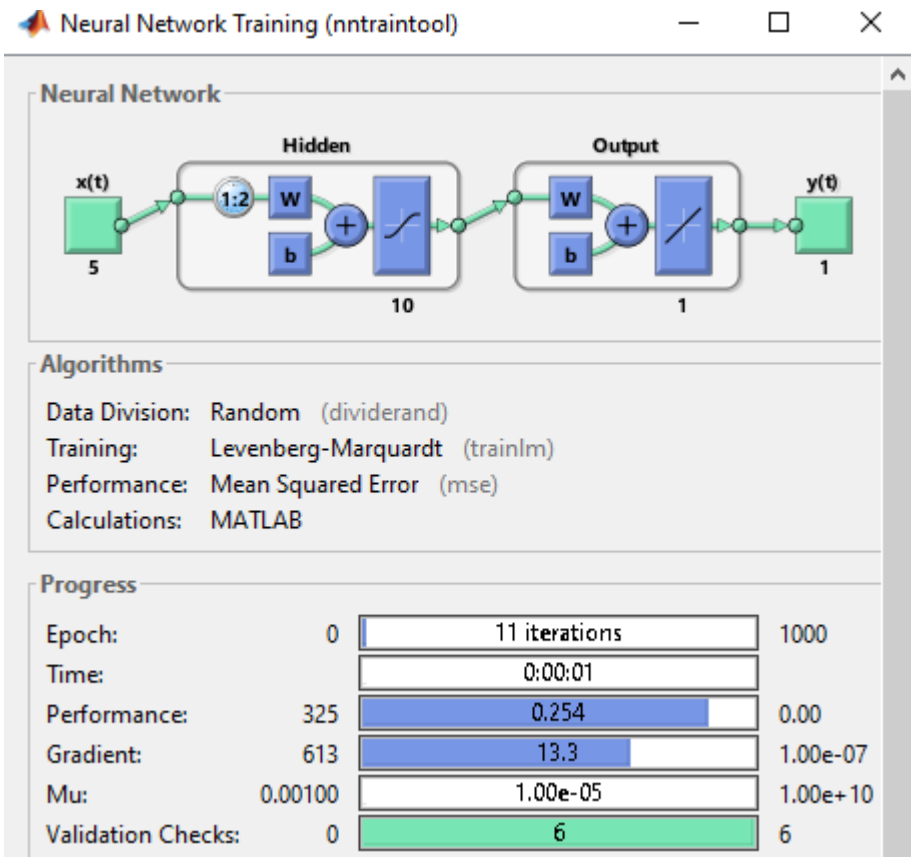


Figure 32. Training window of timedelay neural network.

As shown in **Figure 33**, a great deal of variation was observed between the target and the neural output. The reason for such great disparities in values remains unknown having changed the training algorithm to a more sophisticated training like *trainscg* and *trainrp* respectively.

The output is as given below:

targetOutputs	NeuralOutputs
[3.4762]	[2.9964]
[-21.7538]	[-18.7178]
[3.8645]	[-3.3995]
[-6.9113]	[-8.2706]
[-10.6668]	[-10.8536]
[4.7497]	[-5.2249]
[-15.9633]	[-12.3829]
[-11.2684]	[-7.3215]
[3.0855]	[4.8987]
[-1.9639]	[-0.7786]
[-15.9083]	[-17.2771]
[-5.6676]	[-2.9583]
[-10.0151]	[-9.2551]
[-9.7302]	[-7.2136]
[3.1961]	[1.0343]
[-15.1480]	[-13.0705]
[-7.0526]	[-10.0931]
[-16.4780]	[-9.7138]

Figure 33. Target and neural outputs of timedelay neural network.

Regression analysis plot shown in Figure 34 clearly summarizes the fact that the network failed to effectively learn the relationship.

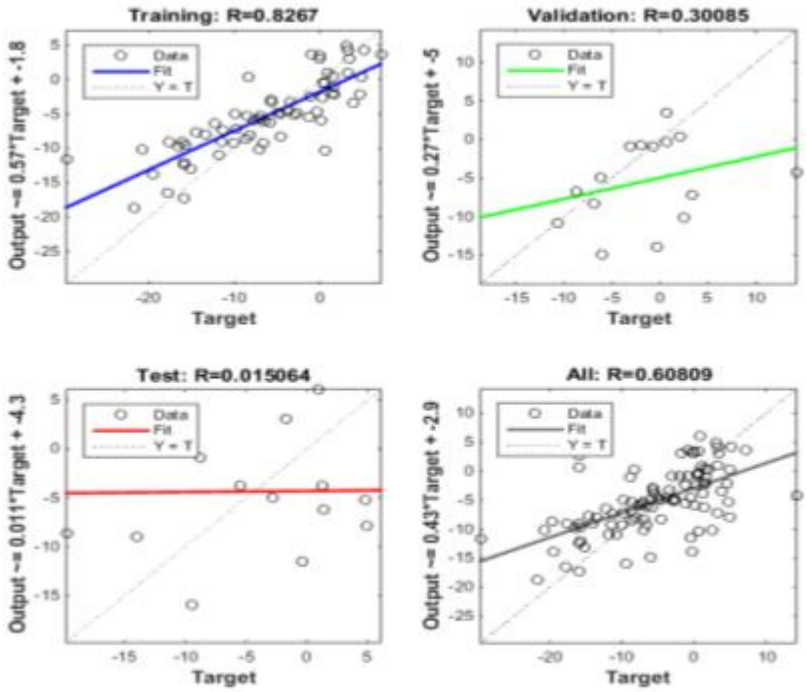


Figure 34. Regression plot of the relationship between the target and the neural outputs.

The regression value R, indicated the degree of the relationship. Finally, the target and the neural outputs are put on a distinct **Figure 35** to clearly have a hint of the variation between the target and the neural output.

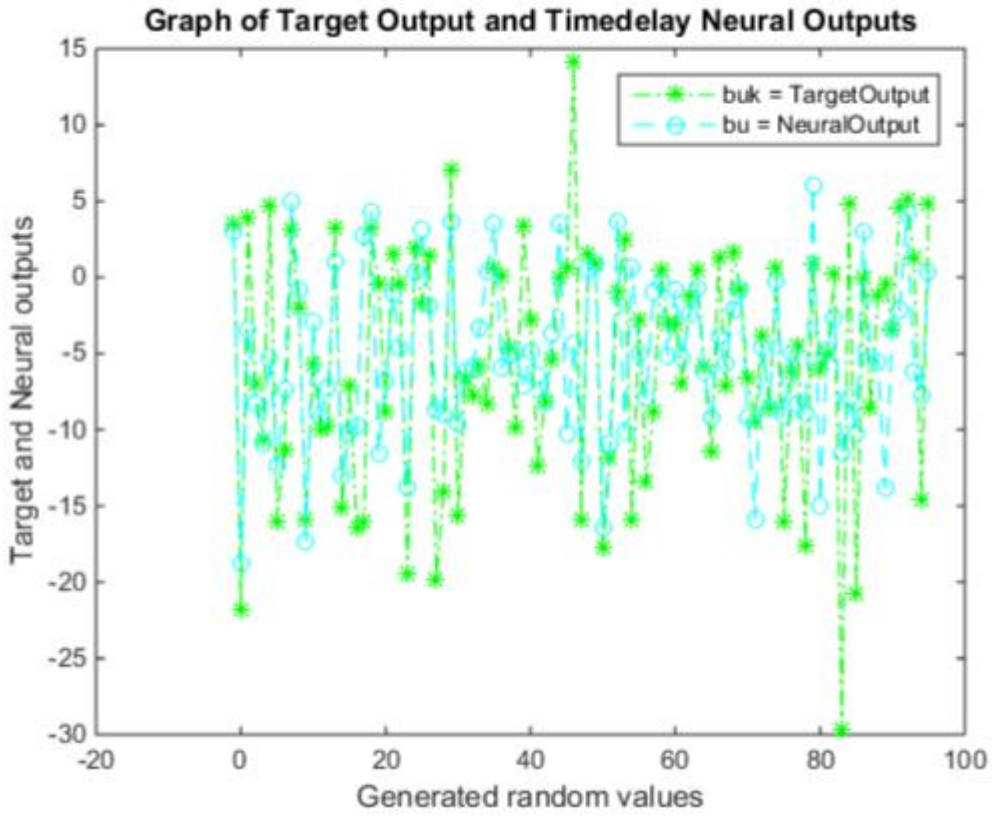


Figure 35. Variation in target and neural output plot.

Though, it is worthy to mention that an improvement in the learning efficiency was observed in terms of the performance and how close the target values were with the neural output when distributed delay neural network, *distdelaynet* was used. This is because it uses delay on the layer weights as well as the input weight. However, this is beyond the scope of this thesis.

4.4 Layer Recurrent Neural Network

Layer recurrent neural network uses the function *layrecnet* and the command window is shown in **Figure 36**.

```
Command Window
>>
>> clear
>>
>> A = (dinputs)';
B = (douts)';
X = con2seq(A);
T = con2seq(B);
>>
>> net = layrecnet(1:2,10);
[Xs,Xi,Ai,Ts] = preparets(net,X,T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
Y = net(Xs,Xi,Ai);
perf = perform(net,Y,Ts)

perf =

   2.0101e-25
```

Figure 36. Command window for layer recurrent neural network.

Similarly, **Figure 37** showed some of the learning outcomes of both target and neural outputs respectively

```
Bold =

    targetOutputs     NeuralOutputs
    _____________     _____________

    [  3.4762]        [  3.4762]
    [-21.7538]        [-21.7538]
    [  3.8645]        [  3.8645]
    [ -6.9113]        [ -6.9113]
    [-10.6668]        [-10.6668]
    [  4.7497]        [  4.7497]
    [-15.9633]        [-15.9633]
    [-11.2684]        [-11.2684]
    [  3.0855]        [  3.0855]
    [ -1.9639]        [ -1.9639]
    [-15.9083]        [-15.9083]
    [ -5.6676]        [ -5.6676]
    [-10.0151]        [-10.0151]
    [ -9.7302]        [ -9.7302]
    [  3.1961]        [  3.1961]
    [-15.1480]        [-15.1480]
    [ -7.0526]        [ -7.0526]
    [-16.4780]        [-16.4780]
```

Figure 37. Target and neural outputs of layer recurrent neural network.

From **Figure 37**, it can be said that layer recurrent neural network perfectively learned the relationship between the inputs and the output.

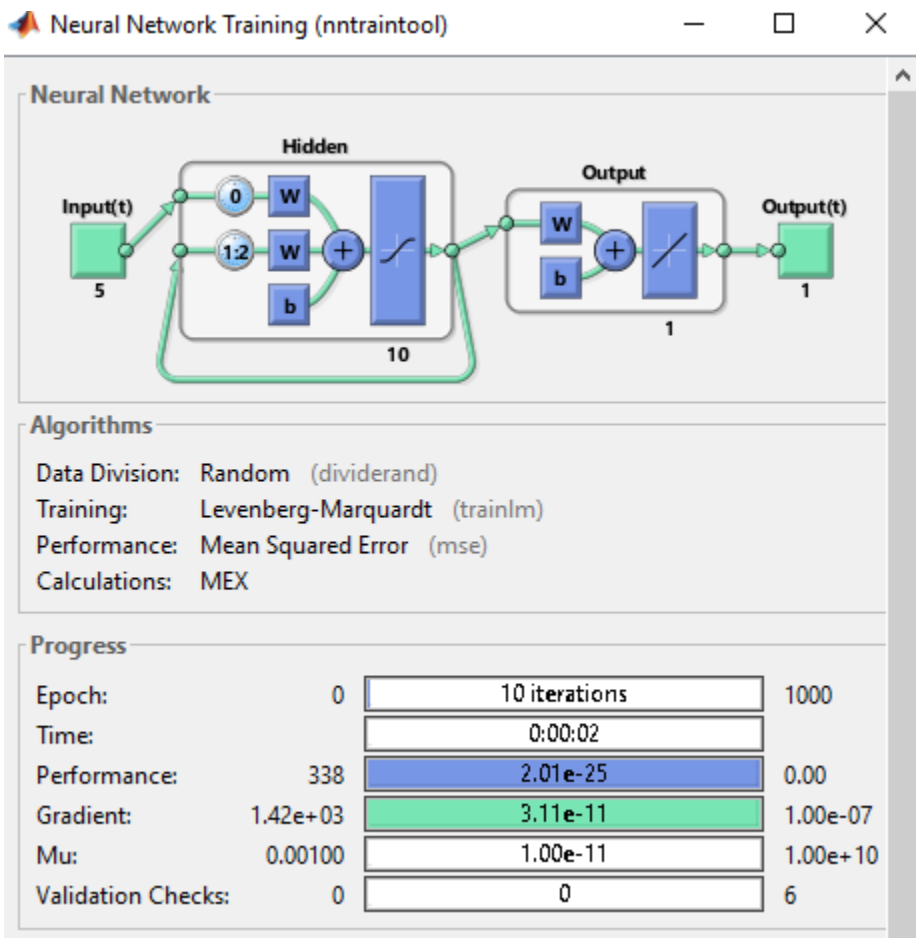


Figure 38. Learning window of layer recurrent neural network.

Levenberg-Marquardt was also used in the training and the dataset was randomly divided. The performance was calculated using the Mean Squared Error.

The quick view plot shown in **Figure 39** gives a quick view of the values in the target and neural output.

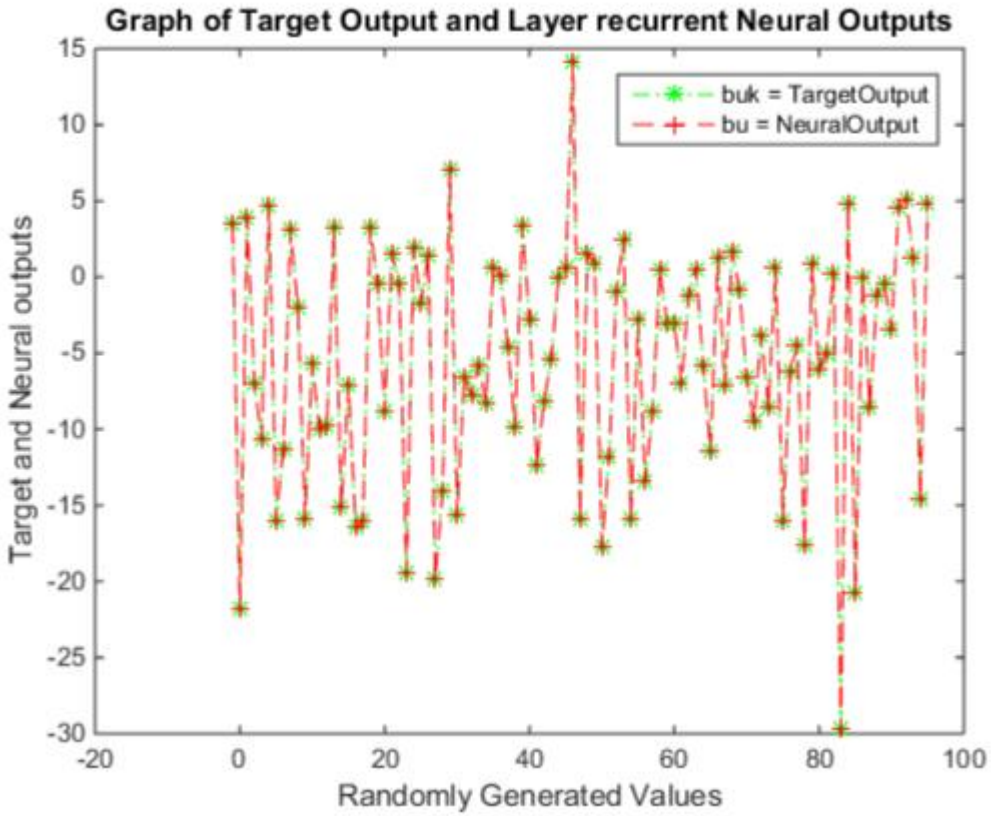


Figure 39. Target and neural outputs of layer recurrent neural network.

From plot obtained in **Figure 39**, it means that the neural network effectively learned the relationship and thus, there was no observable difference between the target and neural output.

4.5 Nonlinear Autoregressive Neural Network (NARNET)

Nonlinear autoregressive neural network is trained using the function *narnet*. The command window for *narnet* is shown in **Figure 40**.

```
Command Window
>>
>> clear
>>
>> A = (dinputs)';
B = (douts)';
X = con2seq(A);
T = con2seq(B);
>>
>> net = narnet(1:2,10);
[Xs,Xi,Ai,Ts] = preparets(net,{},{},T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
Y = net(Xs,Xi);
perf = perform(net,Ts,Y)

perf =

   85.5377
```

Figure 40. Command window for the nonlinear autoregressive neural network.

It is important to examine the performance of narnet as it offers full dynamic derivative calculation. Although, this neural network will not considered in the analysis of data contained in Appendix IV, but it is necessary to introduce this network as it will be recommended to be the neural network for future analysis due to its usage of *fpderiv* algorithm.

As with the previously enumerated neural networks, the dataset was randomly divided.

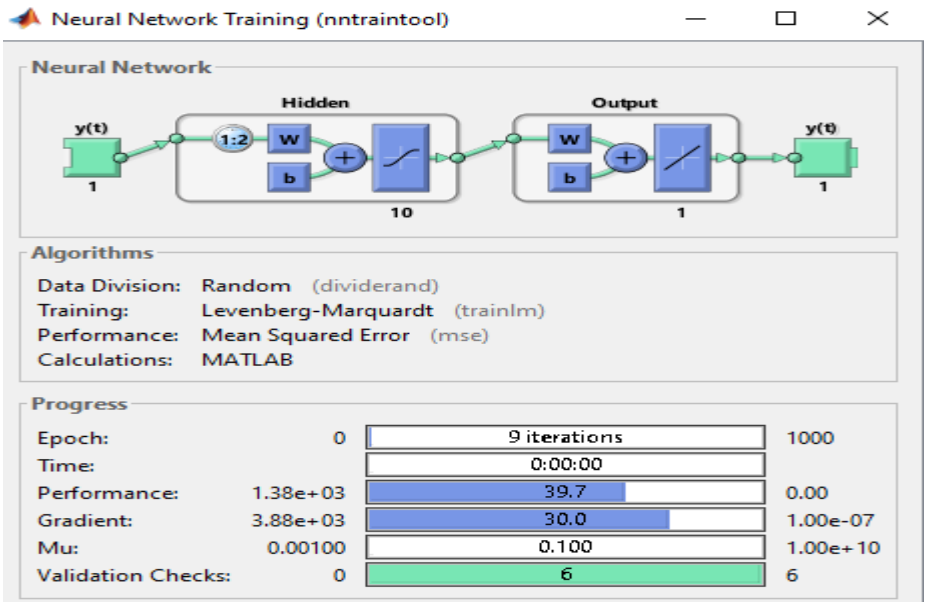


Figure 41. Training window of the nonlinear autoregressive training.

Levenberg-Marquardt was also used in the training. Some of the outputs are as shown in **Figure 41**. From the output, clear disparities in the values of the target and the neural can be clearly seen.

```
Bold =

    targetOutputs    NeuralOutputs
    _____________    _____________

    [  3.4762]       [ -5.4805]
    [-21.7538]       [ -9.7633]
    [  3.8645]       [ -4.0265]
    [ -6.9113]       [  6.9709]
    [-10.6668]       [ -7.3454]
    [  4.7497]       [ -7.7111]
    [-15.9633]       [-10.6605]
    [-11.2684]       [ -8.4451]
    [  3.0855]       [-10.9051]
    [ -1.9639]       [ -8.7663]
    [-15.9083]       [ -7.8621]
    [ -5.6676]       [ -8.2633]
    [-10.0151]       [ -7.2500]
    [ -9.7302]       [ -6.4270]
    [  3.1961]       [ -9.5375]
    [-15.1480]       [ -9.5308]
    [ -7.0526]       [ -7.5112]
    [-16.4780]       [ -9.1413]
```

Figure 42. Narnet training results showing both target and neural outputs.

In addition, the regression plot shown in **Figure 43** indicated that the neural network did not effectively learn the relationships between the inputs and output.

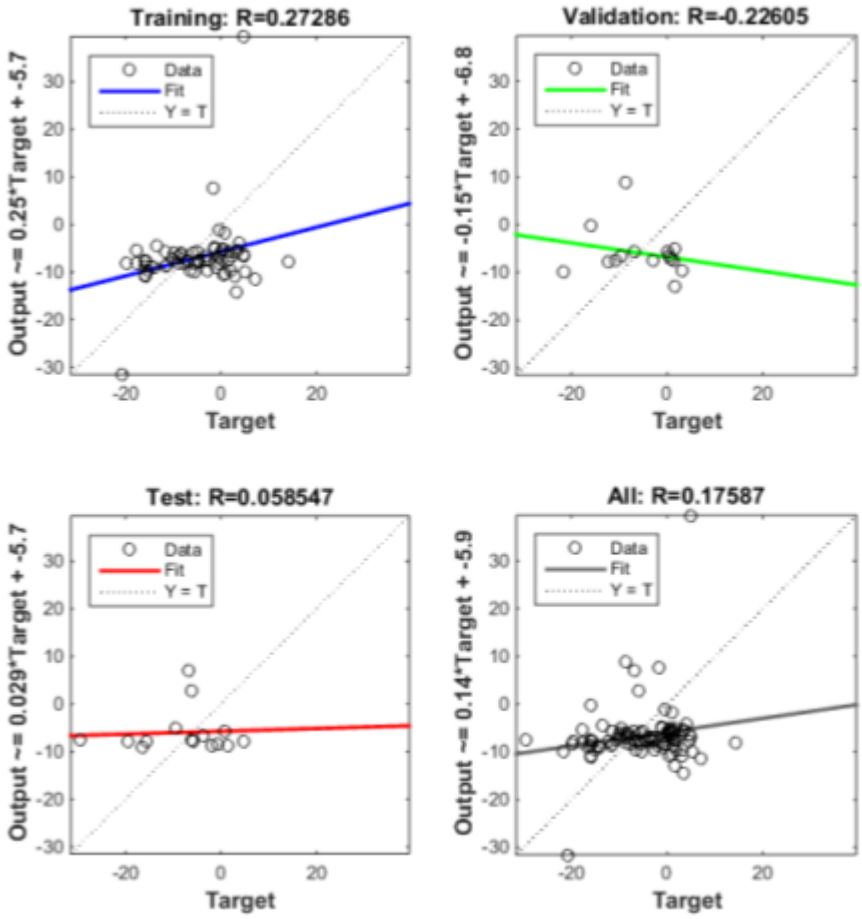


Figure 43. Narnet regression plot of the learning process.

The regression value was R = 0.17587, this value is nowhere less than 0.5. It therefore means that, there was no relationship whatsoever between the target and the neural output.

To put the relationship in a more readable format, **Figure 44** quickly gives an insight into the variation between the expected output and also the trained output respectively.

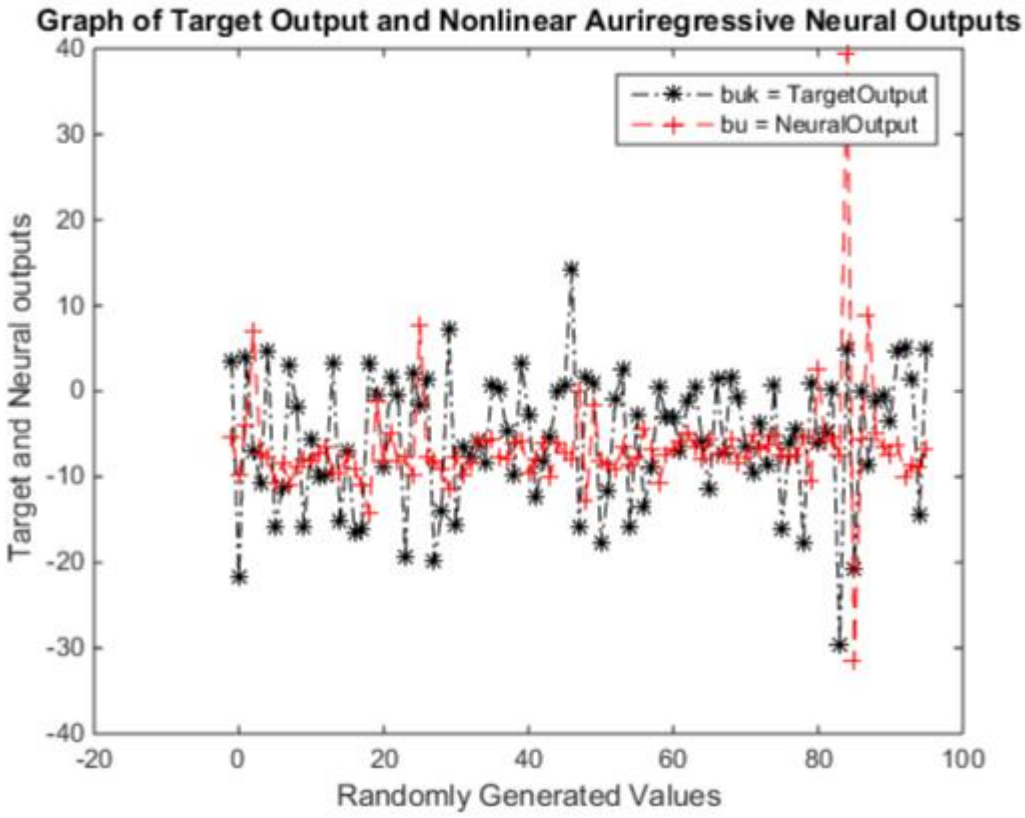


Figure 44. A graph of the target and neural values after training with Narnet.

Since there was no much agreement between the expected value and the trained value. Changing the training algorithm and increasing the number of hidden neurons did not have any meaningful result on the output, the need to examine a variant of the nonlinear autoregressive neural network which is known as nonlinear autoregressive neural network with external inputs. This will be examined in the final section of this chapter, **Chapter 4**.

4. 6 Nonlinear Autoregressive Neural Network with External Inputs

The function used by this network is known as narxnet. This is similar to narnet with the exception of the external inputs. Appendix II was used as the training dataset for this network. Sequel to the use of Appendix II, the MATLAB command window is shown in **Figure 45**.

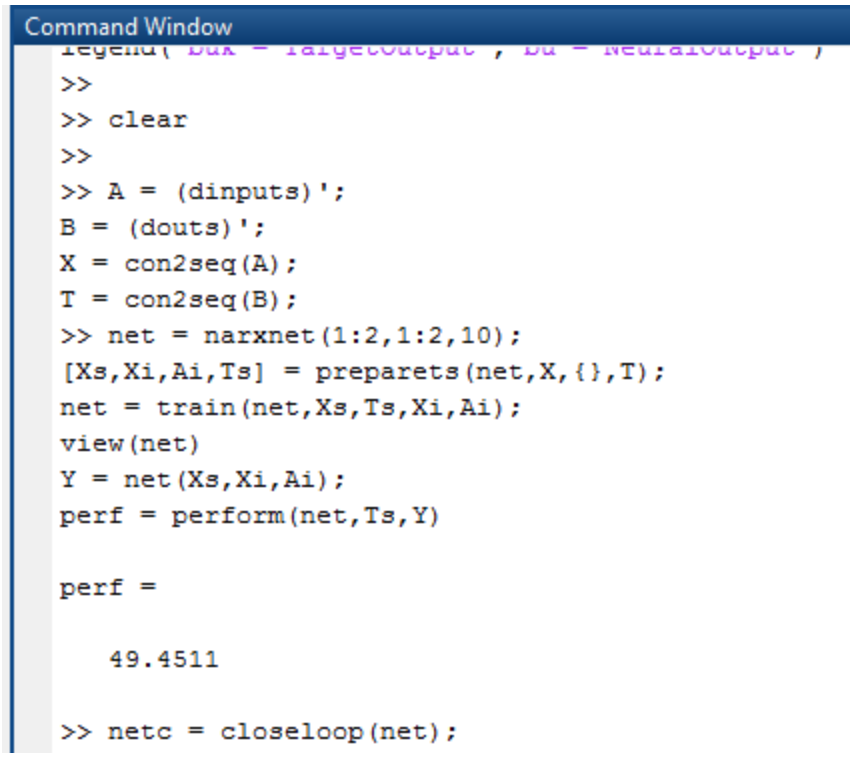


Figure 45. Narxnet MATLAB command window.

Interestingly, narxnet is able to make prediction given the values of the past series, feedback input, and another time series called external or exogenous time series. Hence, the name nonlinear autoregressive neural network with exogenous input. However, this feature is not needed for the purpose of this thesis as this thesis deals with patient's data and the physiological conditions of patients varies from patients to patients.

The training algorithm used was Levenberg-Marquardt and the dataset was randomly divided as shown in **Figure 46**.

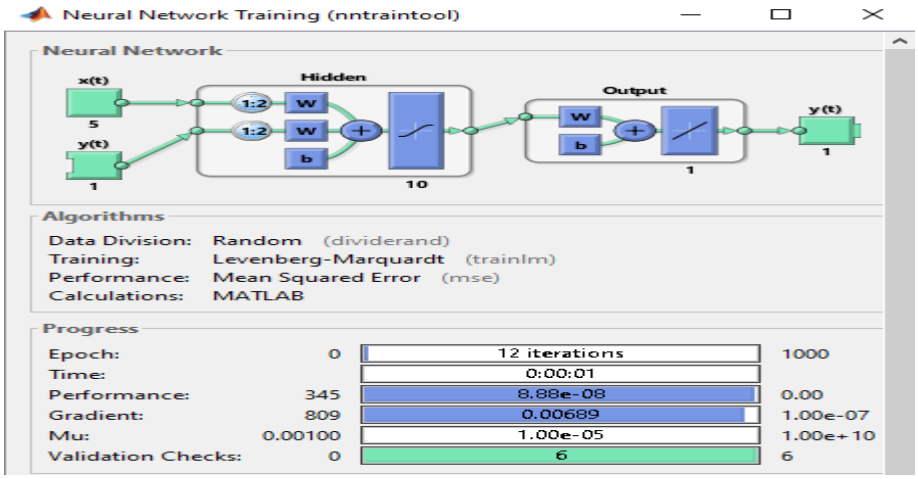


Figure 46. Training window for narxnet.

The target and neural outputs are shown in **Figure 47**.

```
Bold =

    targetOutputs    NeuralOutputs
    _____________    _____________

    [  3.4762]       [ -2.3667]
    [-21.7538]       [  1.3770]
    [  3.8645]       [-10.5209]
    [ -6.9113]       [-10.2902]
    [-10.6668]       [ -7.2775]
    [  4.7497]       [-10.6393]
    [-15.9633]       [-13.9536]
    [-11.2684]       [ -6.0769]
    [  3.0855]       [ -5.7756]
    [ -1.9639]       [ -4.3515]
    [-15.9083]       [ -9.1223]
    [ -5.6676]       [ -1.4770]
    [-10.0151]       [ -2.4262]
    [ -9.7302]       [ -8.5921]
    [  3.1961]       [-11.7863]
    [-15.1480]       [-12.5393]
    [ -7.0526]       [ -8.7021]
```

Figure 47. Narxnet target and neural outputs.

The regression plot in **Figure 48** indicates the inputs-output relationship. The value of the regression plot showed that there was a little or no relationship.

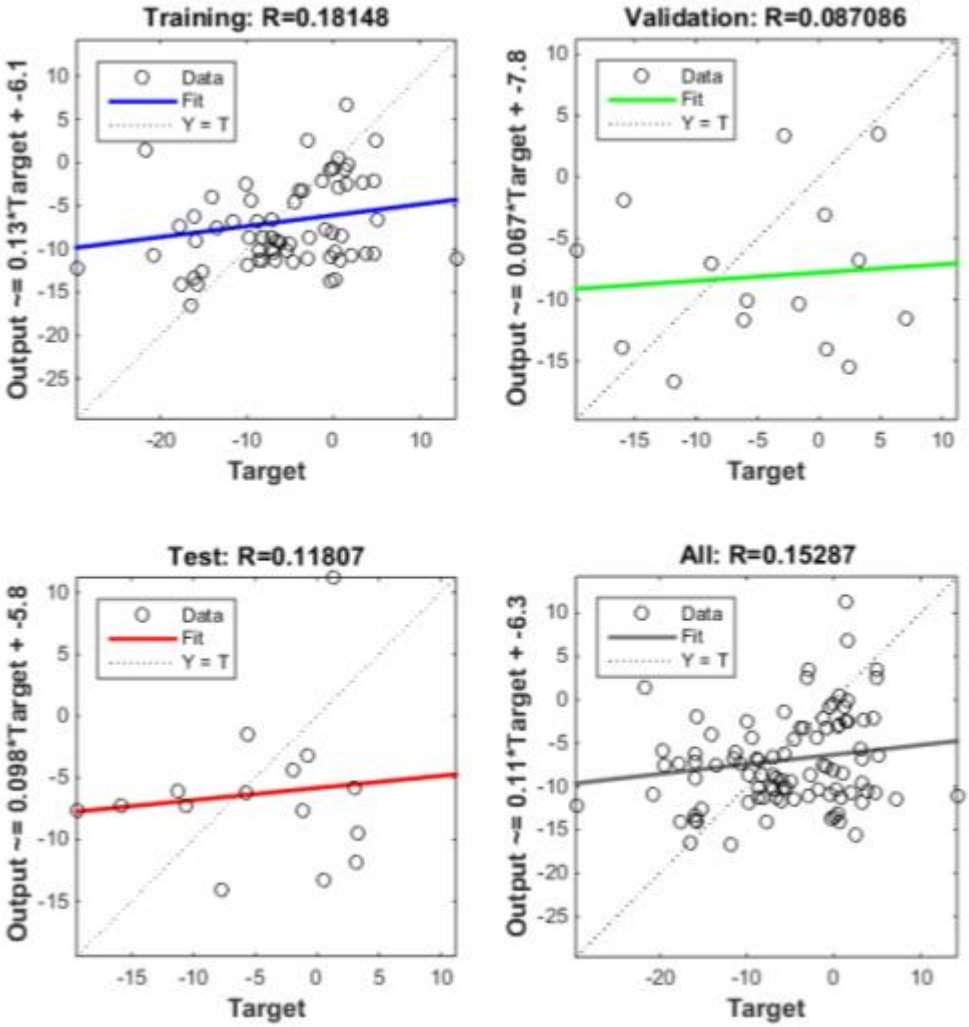


Figure 48. Regression plot for narxnet.

Finally, a quick view of the target and the neural plot is shown in **Figure 49**.

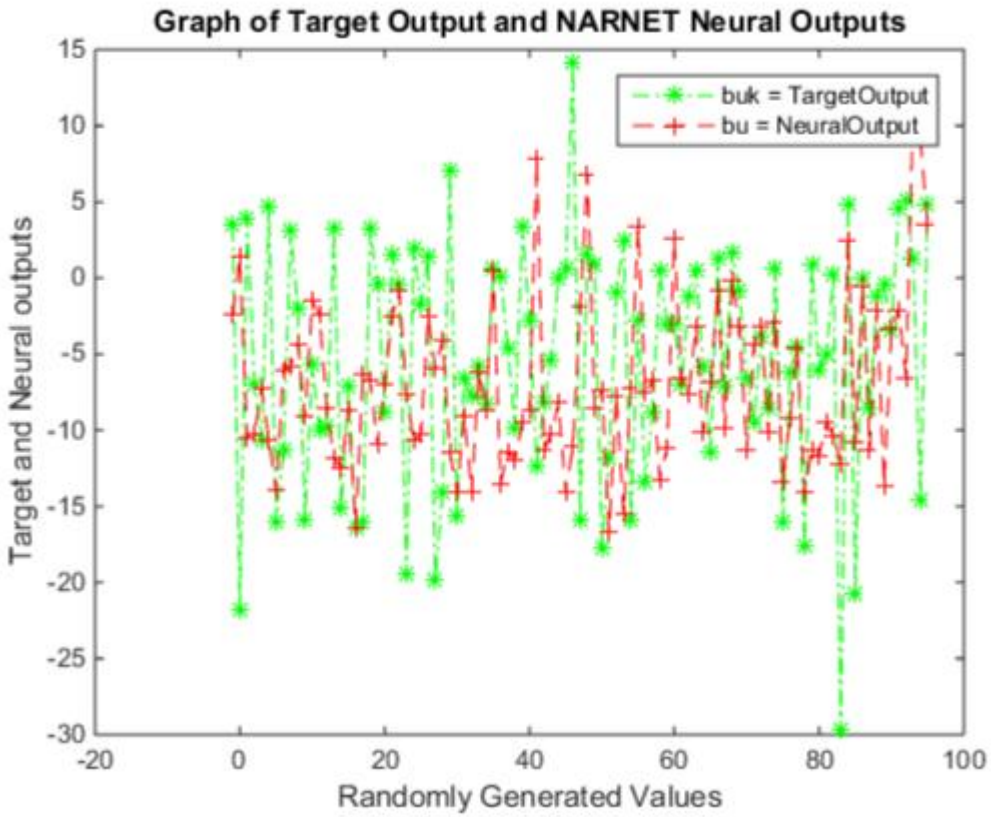


Figure 49. Narxnet target and neural outputs.

In conclusion, based on the use of dataset contained in Appendix II on different neural networks in **Chapter 4** of this thesis, it can be concluded that *feedfowardnet, elmannet and layrecnet* provided a good learning outcome of the relationships between the inputs and output. In other words, the values of the target and the neural outputs are in agreements beyond reasonable doubts. Therefore, these will be the neural networks to be considered for the case study analysis that will be done in the next chapter, that is, **Chapter 5**.

5. ANN SIMULATION OF THE TONGUE CANCER'S CASE STUDY

This chapter is aimed at examining the objectives of the thesis as outlined in **Chapter 1** of this write-up. As pointed out in the introductory chapter of this thesis, cancer is a dreadful disease and the need to be proactive in the diagnosis and treatment becomes imperative. Tongue cancer, that is, oral (mobile) tongue Squamous Cell Carcinoma (SCC) is not an exception. Although, the tongue is characterised by the fact that it has a high amount of muscle bundles which may inhibit the potential tumour spread on it. Despite this fact, detection and prognostic studies are very important in other to record significance success in the crusade against cancer. While detection cancer has to do with having the information that the patient has cancer at an early state, the prognosis on the other hand examines the likelihood of survival of the patients. It is worthy of note that early detection of tongue cancer is not always an indicator for good prognosis (chance of survival from treatment). This is because evidence had shown that about 20% to 40% had spread vigorously to other parts (metastasis) (Ganly et al 2012 & Ho et al 1992).

Thus, prognostic studies are poised to divide the patients into two- firstly, the patients whose tongue cancer is severe and thus would need aggressive treatment such chemotherapy or multimodal therapy. Secondly, the patients who would need surgical treatment alone (Kellermann et al 2007). This important classification into low and high risk will represent a major advancement in the management of this dreadful disease. Clinical size classification into T1 and T2 of early oral tongue SCC was unable to divide the patients into low risk and high risks respectively (Keski-Santti et al 2007). Hence, the need to look out for parameters to be used in the prognostication of cancer becomes imperative. Previous studies had used histomorphological parameters such as depth of invasion, tumour budding, the histological risk to mention but a few (Almangush et al 2013). Most of these histomorphological parameters made up the columns of Appendix III. In addition, the meaning of some of these parameters would be defined in section 5.1.

5.1 Definition of SCC related terms

5.1.1 Tumour Budding

Tumour budding means loss of cellular cohesion as well as loss of active invasive movement. Both of these are malignancy properties.

5.1.2 Tumor size, Prognosis and Metastasis

Prognosis is a measure of survival while metastasis defines the spread of cancer from one part of the body or the affected part to another. Tumor Size is the diameter of the tumor (Baran et al 2015).

5.1.3 Depth of Invasion (DOI)

This is a measure of the thickness of the tumor. In other words, it defines the extent of the growth of the tumour. There are different pattern of invasions with worst patterns of invasions (WPOI) known as *type 4* . There are other types such as *type 5 tumour satellite* and also perineural invasion (PNI).

5.1.4 Symptom

The symptom can be said to be an indicator for a person or patients when the person changes from the normal condition or feelings to an unusual state or feelings due to the presence of something or a disease. In terms of cancer, it can be interpreted as a measure of the aggressiveness of the tumour, which has significant effects on prognosis (Baran et al 2015).

5.1.5 Pathological Stage (TNM Stage)

Pathological stage cTNM classifies in terms of the tumour size and location. TNM is the nowadays the commonly used pathological staging and it is defined below:

T size and location of the main tumour,

N describes the extent of spread to nearby (regional) lymph nodes. Usually, cancer often spread first to the lymph nodes. The lymph nodes are small bean-sized collections of immune system cells.

M indicates whether cancer has spread to other parts of the body.

5.2 ANN in clinical prognostication

Histomorphological parameters have been used in the previous studies because of their relevance on cancer and especially on tongue cancer. Furthermore, evaluations of these parameters are easy and they are also practicable. Several models have been proposed to differentiate patients into low and high risk respectively. An example of such model was the model proposed by Jakobsson et al, Anneroth et al, Bryne et al, and Martınez–Gimeno et al. However, most of these models are found to be complex, therefore they can't be considered in clinical diagnostics. In addition, most of these models have not shown enough prognostic significance especially for oral SCC (Po Wing et al 2002; Silveira et al 2007; Weijers et al 2009 ; and Brandwein–Gensler 2010). Alhadi et al 2013 and other groups such as Kellermann et al 2007 and Marsh et al 2011 have described a strong association between the density of cancer-associated fibroblasts (CAFs; increased a-smooth muscle acting [α-SMA] immunostaining) and a higher mortality in oral tongue SCC and oral SCC.

Therefore, this chapter aimed at using ANN in the prognosis of cancer. Re-occurence of cancer which is called recurrence is an important factor in cancer management. The need to have certainly proven metrics on the recurrence nature becomes imperative. Usually, recurrence is normally predicted using chats such as nomograms and statistical analysis tools. Most of these approaches are designed for 1-5 years recurrence prediction after treatment and also they are sometimes not effective. Therefore, the need for another approach to effective recurrence prediction cannot be over emphasized. In this study, an alternative method based on ANN is designed, to predict the probability of freedom of recurrence for tongue cancers. The tongue cancer patients dataset is contained in Appendix III. Therefore, ANN was used to analyze the variables contained in the dataset of Appendix III. ANN outcome is poised to ease the physician's post-operative follow up procedures. The parameters affecting the recurrence are the TNM stage, tumor size and nuclear (Fuhrman) grade, the existence of necrosis and vascular invasion.

5.3 Data collection and training process

The description of the dataset used in this thesis was given in section 1.1. The data were collected in 5 centers, namely- University Hospitals of Helsinki, Oulu, Turku, Tampere, and Kuopio between 1979 and 2009. About 340 diagnostic histological slides of patients were collected. The criteria for inclusion were as described by Alhadi et al 2013 (Alhadi et al 2013). After analysis, few patients failed to meet the criteria set by Alhadi et al 2013. The data was later arranged and saved in both excel and SPSS formats and handed over to me for analysis using ANN. As this is a further research on the earlier work of Alhadi et al, the approval for the use of the data for research and academic purposes was granted by the University Hospital Ethics Committees of all 5 hospitals and by the National Supervisory Authority for Welfare and Health (VALVIRA). The process of training is as described in **Figure 52** below.

The dataset had been extracted by Alhadi et al (Alhadi et al 2013) and saved in Microsoft Excel format. The columns contained different histomorphological parameters as shown in Appendix III. Also, the training phase usually begins with dividing the dataset into training, validation and testing datasets respectively. This is usually done manually. It is a traditional practice to divide the data into 70% training data and 30% test (15% validation and 15% testing). However, the latest version of MATLAB could automatically divide the dataset into training, validation, and testing. Thus, there was no need for manual division. The dataset used in the training of the ANN for this thesis was divided automatically. Following the division of the dataset are the network creation and configuration. Then, the weights and biases are initialized. Sometimes, the number of hidden neurons can be increased to provide a better network performance. Having divided the dataset, creation and configuration, initialization of weights and biases, the next step is to train the network as shown in **Figure 50**. There are numerous training algorithms for the artificial neural network.

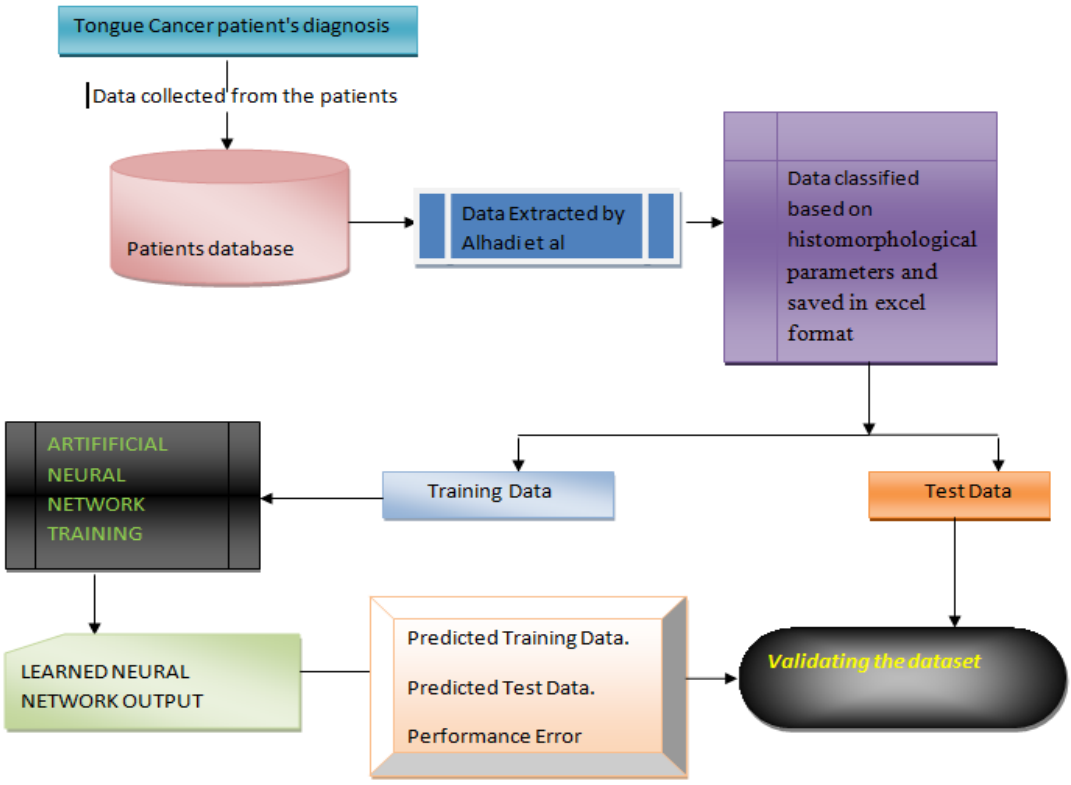


Figure 50. Data collection and training process.

These include Levenberg-Marquardt, BFGS Quasi-Newton, Resilent Backpropagation, Bayesian Regularization, Gradient Descent with Momentum, Gradient Descent algorithm to mention but a few. Each of these algorithms has a readymade function in MATLAB. Levenberg-Marquardt is characterized by *trainlm* while the training function for Quasi-Newton was *trainbfg* they are both fast training algorithms. Furthermore, *trainlm* is the default training for feedforward network. It was the training method used for feedforward network and layer recurrent network as described in **Chapter 4**. Similarly, for Elman neural network, the training method was changed from Gradient Descent with Momentum to Levenberg-Marquardt for better performance error.

5.4 Neural Network for predictions

This section is aimed to answer the objectives of the thesis. Therefore, objectives would be examined in two sub-sections. These are (i) ANN for predictions of recurrence and status of the patient (mortality). (ii) The dependencies of each variable on the target outcome. Each of the variables used are as defined in **Table 7**.

Table 7. Explanation of inputs and outputs variables.

Variable Name	**Explanation**
Blokki	This gives the block code of the patient.
Grade	World Health Organization (WHO) grading based on differentiation
Grade_2	Modified Grade Column
Status	This is the same as Mortality
Budding	Individual cell or cluster of cells at the invasive font of carcinoma.
Budding_2	Regrouping of the values obtained in Budding column.
Depth	This is the depth of invasion

Depth_2	Regrouping of death of invasion
Modified	Clinical staging modified by incorporating depth of invasion.
Timeinmonths	Time of surgery to last follow up or death in months.
Age	Age of the patients
Age_2	Regrouping of the age
Stage	Stage of the cancer
Gender	The masculinity or feminist of the patient
Centre	The centre that the patients belongs to.
Recurrence	The notice of tumour forming after treatment
Diseasefree	Time in months from surgery to recurrence
WPOI	Worst Pattern of Invasion
WPOI_2	Worst Pattern of Invasion
LHR	Lymphocytic host response
LHR_2	Regrouping of LHR column.
PNI	Perineural Invasion
PNI_2	Regrouping of perineural invasion.

The **Table 7** above gives the definition of some of the variables. Some of the variables have been regrouped for simplicity. These would be explained as follows. For instance, *Grade* is based on degree of differentiation as defined WHO. Where "1" means it is well differentiated, "2" means it is moderately-differentiated and poorly differentiated was represented by "3". In addition, *grade_2* grouped the degree of differentiation into two main group. Group 1 for well differentiated while group 2 for poorly differentiated respectively. Furthermore, in the *modified* stage column, clinical staging modified by incorporating depth of invasion. 1=stage I; 2=stage II. Also, disease free time in months from surgery to recurrence. Lymphocytic host response (LHR) has been broadly divided into three types.

In LHR, type 1 means that the LHR is strong. The lymphoid nodules are at an advancing edge. Similarly, type II means that it is intermediate. That is, the lymphoid nodules in some but not all. Lastly, type III indicates that it is weak. Deductively, no lymph nodule. All these variables are the inputs to be fed into the network as shown in **Figure 53**.

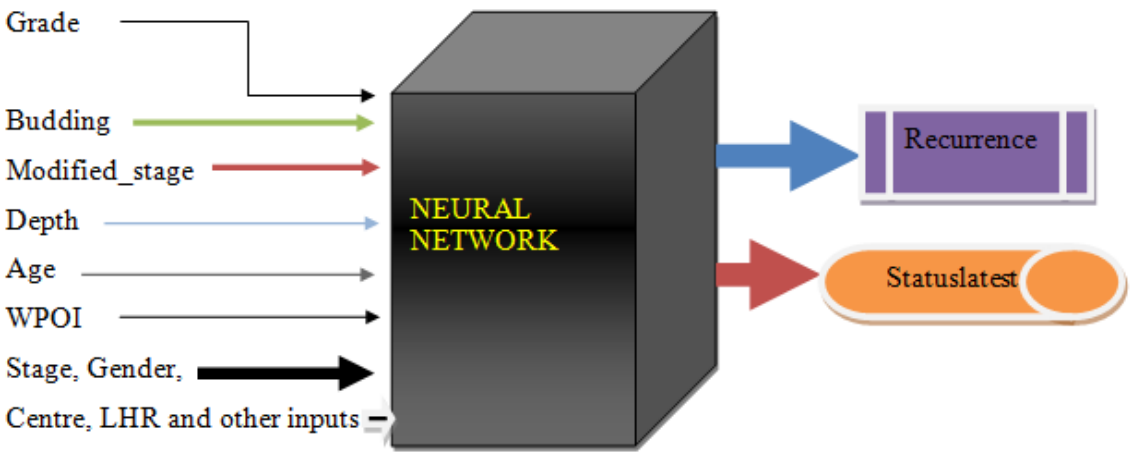


Figure 51. The design of the network with the desired inputs and outputs.

Perineural invasion (PNI) was originally grouped into '1' for none, small tumour wrapping around nerves as '2' and large tumour wrapping around nerves was denoted by value '3'. For simplicity, the *PNI* was re-grouped as *PNI_2* where "0" represent no presence of tumour wrapping around nerves and "1" for the presence of tumour wrapping around nerves. Worst Pattern of Invasion (WPOI) has been largely categorized into three types. Type 1 represents pushing border and was denoted by value 0. Type II was a finger like growth and denoted by value 1. Type III was a large separate islands denoted by value 2. Small tumour island case was type IV and it was denoted by value 3. Thus, for easy classification and analysis, WPOI was re-grouped as WPOI_2 with values 0 and 1 respectively. Where value "0" indicated pushing border while "1" represents other type of growth. Tumour *budding* is an expression of two properties of malignancy's properties. These are loss of cohesion and active invasive movement. To put it succinctly, it represents the presence of single tumour cells or cluster of cells.

With regards to tumour budding column in **Appendix III**, the single tumour cells or cluster of cells have been measured. The value of the measured tumour has been given in the budding column. In the same vein, *budding_2* column is a re-grouping of the values measured in the *budding* column, where budding measurement of less than or equal to 4 (< = 4cm) is grouped as value '0' and greater than 4 (>4) is grouped as '1'. In addition, the depth of invasion or tumour thickness is a measure of invasiveness of the tumour. It has been measured and recorded in the column that was known as *depth*. Also, the column *depth_2* is the re-grouped version where depth of greater than or equal to 4 millimetre (>= 4mm) was grouped with value '1' and less than 4 millimetre (< 4mm) was grouped as '0'.

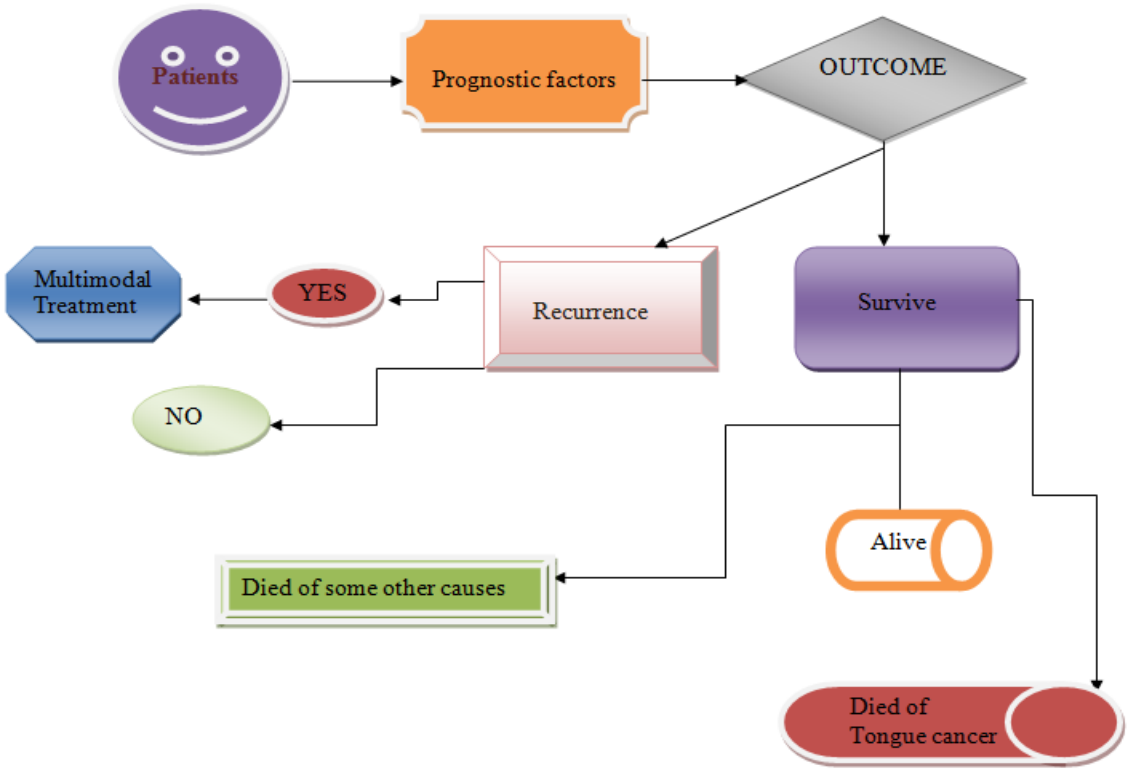


Figure 52. Schematic diagram of the outputs.

In terms of the interested outputs from the variables as contained in the columns, *recurrence* and *statuslatest* are the columns of interest as shown in **Figure 52**. Recurrence is a marker that represents the re-occurrence of the cancer after the initial treatment.

The initial treatment could be normal clinical treatment or aggressive treatment like chemotherapy or radiotherapy as the case may be. The recurrence column was represented by values 0 and 1. Where value ‘0’ means that there was no recurrence and '1' indicated recurrence after the initial treatment. Similarly, survival block in **Figure 52** represent the *statuslatest* also known as mortality. In *statuslatest* in **Appendix III**, value 0 represent that the patients is alive, 1 for died of cancer and value 2 meant that the patient died of other causes.

5.4.1 Prediction of recurrence and mortality using feedforward neural network

Sequel to the simulation of feedforward neural network provided in **Chapter 4** of this thesis, the dataset provided as contained in **Appendix III** and also based on the design architecture in **Figure 51**, the network is trained using feedforward neural network. The aim of the training is to use the trained network for prediction of any new input. In addition, the value of the neurally trained network (neural output) will also be compared to the expected output. In this analysis, the columns were re-arranged and the column *blokki* was totally removed. The modification to the original dataset was saved and labelled as **Appendix IV** which now contained 311 rows and 20 columns [311x20].

The inputs in **Figure 51** above basically contained ***Grade, Grade_2, Budding, Budding_2, Depth, Depth_2, Modified, Time in months, Age, Age_2, Stage, Gender, Centre, Disease free, WPOI, WPOI_2, LHR, LHR_2, PNI and PNI_2*** respectively. While ***Recurrence and Status latest*** are outputs respectively. For easy analyses, outputs are treated separately. That is, recurrence was considered as an output separately. Similarly, mortality was considered separately.

For **recurrence** as output:

```
>> clear
>>
>>
>> x = (dinputs)';
>> t = (douts)';
>> net = feedforwardnet(10);
net = train(net,x,t);
view(net)
y = net(x);
perf = perform(net,y,t)

perf =

    0.1034

>>
```

Figure 53. The command window showing the codes for feedforwardnet.

From **Figure 53**, the performance value was very promising and it showed that the network had effectively learned the relationship. Therefore, **Figure 54** showed how effectively the network had learned after the training process.

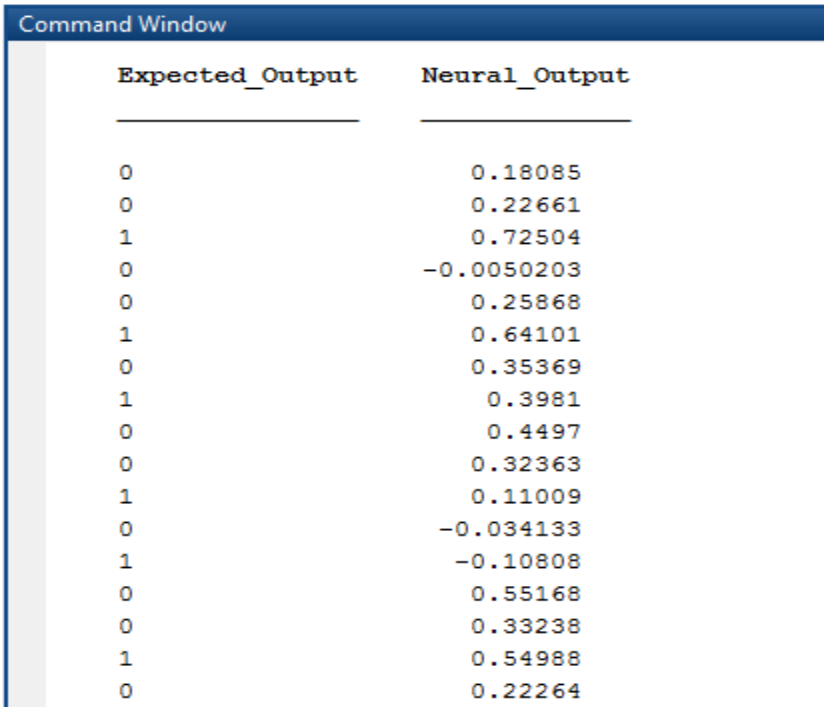


Figure 54. The expected output and the neural output after training with real dataset

A cross section of the output is shown in **Figure 54.** As the data contained about 311 rows, all outputs can't be shown. Some outputs are definitely missing shown is a truncated output. It was aimed to show how the trained network had effectively learned the relationship.

To have an insight into the behaviour of the learning, that is, of how many of the trained value are not exact with the expected value, **Figure 55** provided the extent of variation.

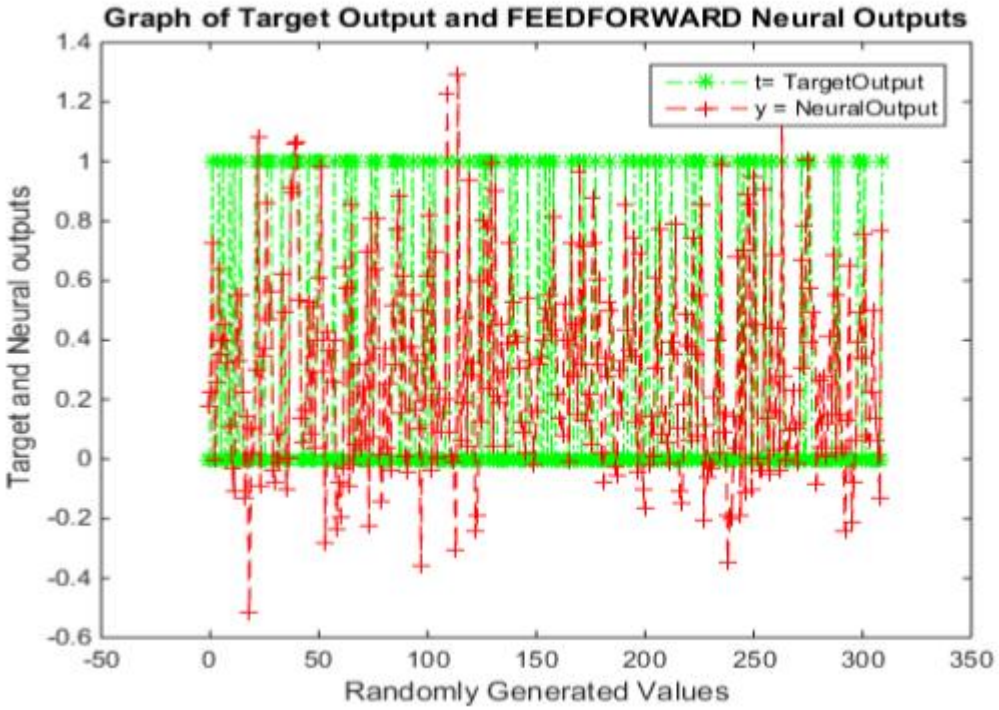


Figure 55. A plot of the extent of variation between the target and neural outputs

The summary of the MATLAB training exercise is given in **Figure 56**.

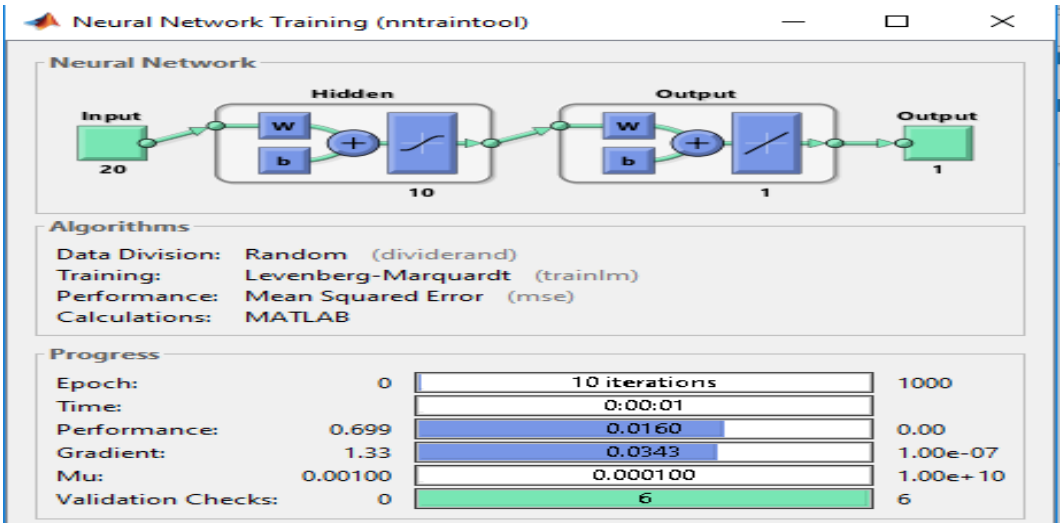


Figure 58. Training summary for feedforward neural network of the real data

Similarly, the regression analysis plot is also a good marker to show the level and extent of agreement between the target and trained output respectively.

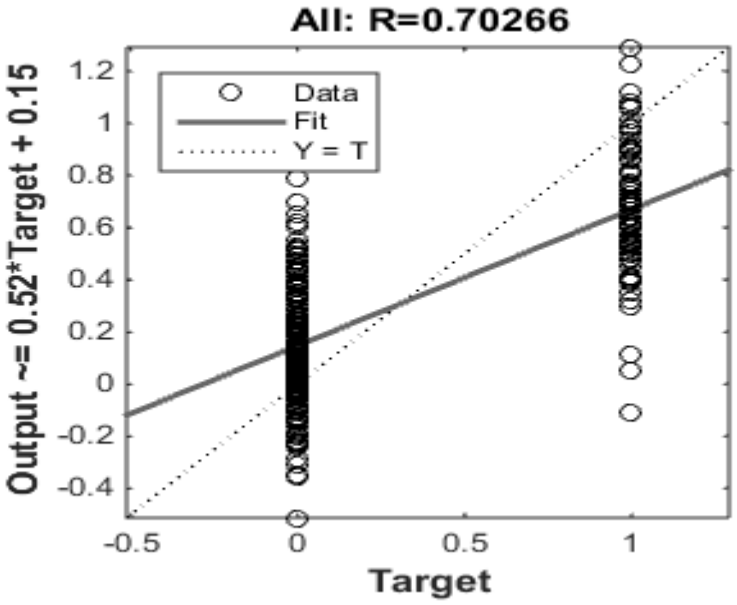


Figure 57. Regression plot from the feedforward neural network of the real dataset.

Since the regression value obtained is much closer to 1, therefore, it can be said that the value obtained from the training can be accepted.

5.4.2 Prediction of Recurrence from Feedforward Neural Network

Two approaches to using the network for prediction were adopted. Firstly, one of the rows in **Appendix IV** was randomly chosen. In this case, the expected recurrence is known. Secondly, an arbitrary input was drafted to see how well the network can make prediction or recurrence. For the first approach, the same input as **row 243** in Appendix IV was fed into the network. Based on the inputs provided in row 243, the output of recurrence is expected to be '**0**', while the recurrence output from the trained network is expected to be -0.20135 as shown below.

```
     0                    -0.18984
     0                    -0.20135
     0                     0.14211
```

However, using the prediction algorithm, the value obtained is as shown in **Figure 58**.

```
>>
>> pred_feedforward = net([[3;2;9;1;1;4.5;2;245;58;1;2;1;6;245;1;1;3;1;1;1]])

pred_feedforward =

   -0.2014
```

Figure 58. Prediction of recurrence as an output given new inputs.

Therefore, there was a strong agreement between the expected recurrence output from the trained network (-0.20135) and the value obtained when the prediction algorithm was used. In both cases, the value of recurrence was less than 0. Thus, it was easy to conclude based on the inputs that ***there was no recurrence (<=0)***. Hence, in other to have a conclusive point on the prediction, a new set of input was formed off hand. The new inputs formed are:

[2;3;8;1;1;3.5;240;57;1;1;1;6;243;1;1;2;1;1;1]

The set of inputs value shown above corresponds to each prognostic factors given in **Appendix IV** of this thesis. This was then used to test the trained network for recurrence classification.

Using the new set of inputs above on the trained network, the output produced is given in **Figure 59**.

```
>> pred_feedforward_newinputs = net ([[2;3;8;1;1;3.5;2;240;57;1;1;1;6;243;1;1;2;1;1;1]])

pred_feedforward_newinputs =

   -0.1131

>>
```

Figure 59. Prediction of recurrence based on newly formed inputs

Interestingly, the output produced was -0.1131 and since the value is less than 0, it means that there would be no recurrence of cancer if such type of inputs values is obtained from patients. It is worthy of note that the number of inputs to be used in the prediction should be the same as the number input that the network was trained with to avoid errors.

5.5 Prediction of Statuslatest using Feedforward Neural Network

As pointed out in the introduction to **Chapter 5**, *statuslatest* column represents the survival column of the patients. Hence, *statuslatest* is otherwise known as mortality. Having said that, it is imperative to mention that both *recurrence* and *mortality/statuslatest* were excluded from the inputs in the previous analysis in Section 5.4. The training summary using *statuslatest* as output is presented in **Figure 60**

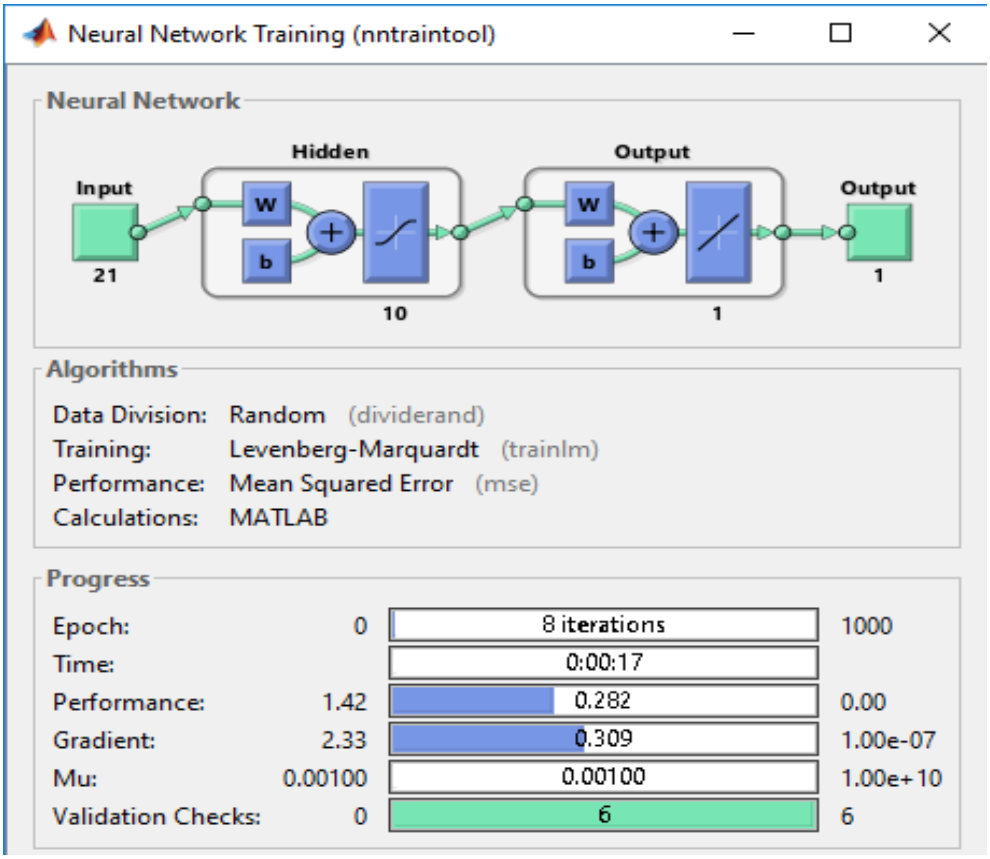


Figure 60. The training summary for statuslatest as output

Since the recurrence can easily be predicted. It means that the recurrence values are known. Thus, it can be said deductively that it can be included as parts of the inputs to be fed into the feedforward network. Hence, the output is this case is *statuslatest*.

Therefore, the **Figure 61** showed the MATLAB input command. It showed the workspace code and error performance for mortality(statuslatest) as the output.

```
Command Window
>> x = (dinputs)';
t = (douts)';
>> net = feedforwardnet(10);
>> net = train(net,x,t);
view(net)
y = net(x);
perf = perform(net,y,t)

perf =

    0.6897

>> bold = table (t',y','variableNames',{'Expected_Output' 'Neural_Output'})
```

Figure 61. Command window code for statuslatest as output.

With regards to the value of the trained network and the target/expected value, the output is given in the **Figure 62**.

```
Command Window
>> bold = table (t',y','variableNames',{'Exp

bold =

    Expected_Output    Neural_Output
    _______________    _____________

    2                    1.0461
    0                   0.92631
    1                    1.6794
    2                      1.44
    0                    1.2954
    0                    1.1857
    2                         1
    0                   0.55464
```

Figure 62. Expected and Neural outputs where mortality was the output variable.

Based on the output given in the **Figure 62**, it became evident that the network could not effectively learn the relationship between the inputs and mortality. Therefore, the need to examine all the output of the trained network against the expected outputs becomes imperative. Hence, the **Figure 63** gives a brief overview.

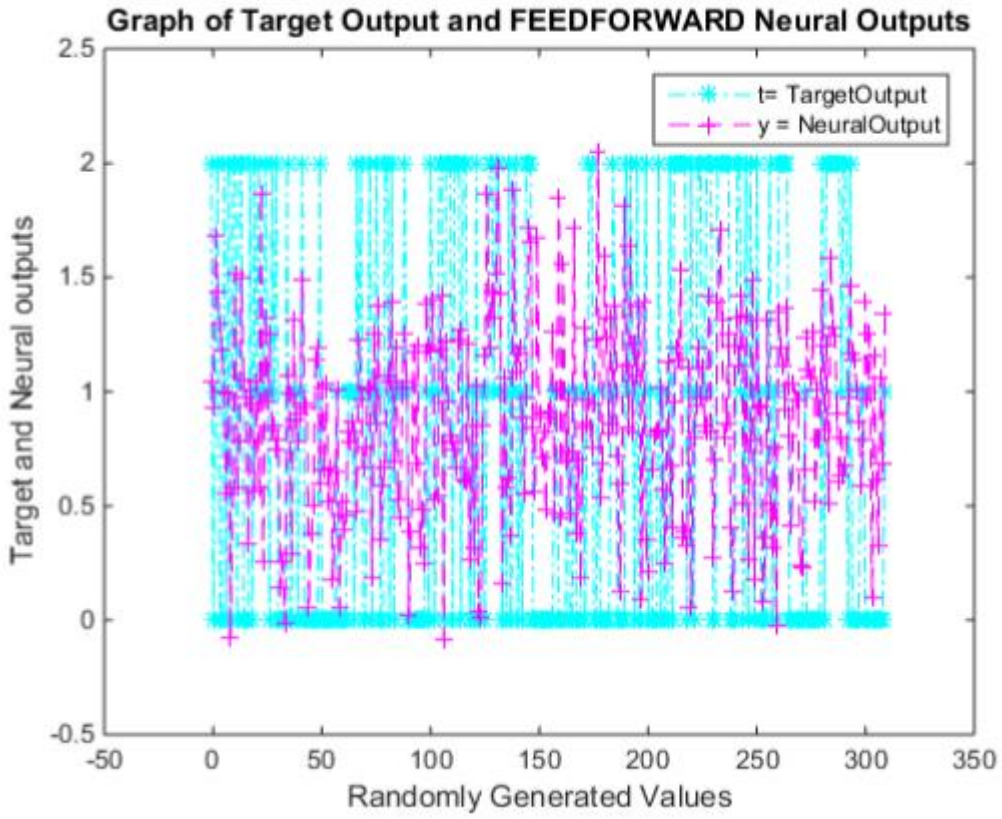


Figure 63. The representation of the target and neural output where statuslatest was the output.

From **Figure 63**, the level of the disparities in the expected and the neural output could clearly be seen. The extent of the disparity calls for concern as it becomes evident that the network might not be effectively predicting the mortality.

Having said that, it is worthy of note to examine the training performance and the regression plots for mortality as output. These plots would help to finally determine if the learning phase was successful or not.

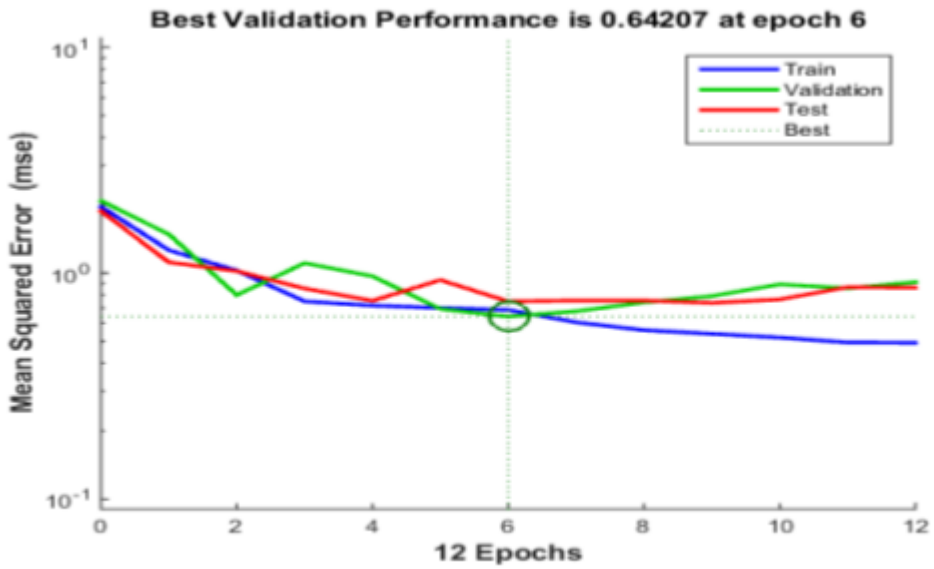


Figure 64. The feedforward network performance when mortality was the output

The training, validation, and testing had the best fit at the 6th epoch. It means the performance would not be effective.

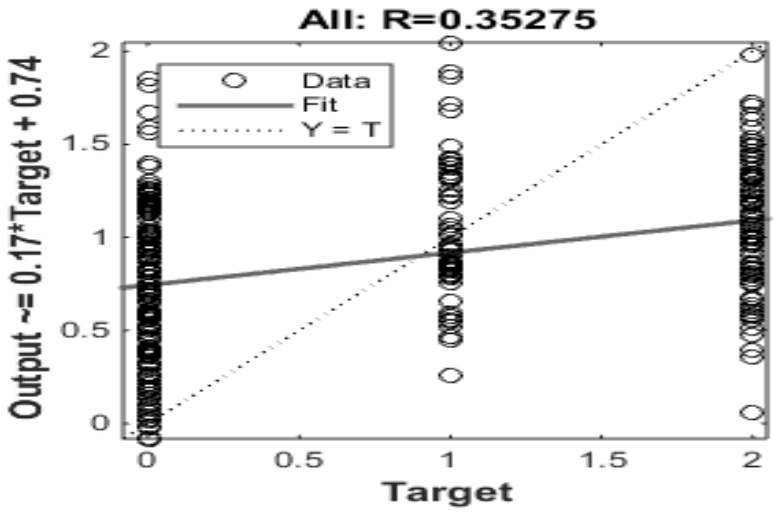


Figure 65. Regression plot for mortality as output using feedforward network

To further confirm this, the regression plot was also examined as depicted in **Figure 65**. From **Figure 65**, it became evident that the value obtained was ***0.35275*** which was less than 1. Therefore, it can be deduced from the regression plot that the network did not effectively learn the relationship between the inputs, target and neural outputs respectively. Thus, the need to examine another type of neural network presented in **Chapter 4** becomes imperative. However, the training algorithm can also be changed to other fast algorithms like conjugate gradient algorithm or resilient backpropagation to examine if the network would be able to learn the relationship between the inputs and the output.

5.5.1 Prediction of Mortality using Feedforward Neural Network

Controlled and uncontrolled prediction approaches are used to make predictions. By controlled predictions, it means one of the rows of the inputs is used. It is controlled because the output is known already. This is to avoid unnecessary error in the interpretation of the results. In this case, the row 310th was randomly chosen. The inputs contained in the row 310th are: **[3;2;2;0;0;3.0;1;33;50;1;1;1;6;33;1;1;3;1;0;0;0].** The expected value '0' and neural value '0.68632' are given below:

```
0          1.0596
0         0.68632
1          1.3437
```

From the trained network, the prediction of the given inputs is given in **Figure 66**.

```
>> pred_feedforward_mortality = net([[3;2;2;0;0;3.0;1;33;50;1;1;1;6;33;1;1;3;1;0;0;0]])

pred_feedforward_mortality =

    0.6877
```

Figure 66. Controlled prediction using one of the known inputs row.

The disparity between the expected and the neural output would not make the results to be reliable. Although the value from the trained network agreed perfectly with the predicted network, yet the result is not reliable. This is because the network did not learn properly. As presented above, it is expected that the with the input, the output of mortality is expected to be '0' which means the patient is *alive*. However, from the predicted outcome, the value was 0.6877 which could be approximated to 1. Hence, it would therefore means that for the inputs, the conclusion would therefore mean that the patient died of *cancer*. Obviously, this is not the case. Similarly, an uncontrolled prediction means that the set of inputs was just formed arbitrarily and given in the **Figure 67**.

```
>> pred_feedforward_mortality_newInputs = net([[2;2;6;0;0;17.2;1;5;50;1;1;1;6;33;1;1;3;1;0;0;0]])

pred_feedforward_mortality_newInputs =

    1.2925

fx >>
```

Figure 67. Uncontrolled predictions of arbitrary inputs.

From the results of the uncontrolled predictions in **Figure 67**, it is difficult to make any meaningful predictions because there were no reasonable correlations between the expected output and the trained output. Based on this, it is imperative to change the training algorithm to examine if that would assist the network to learn the correlation between the inputs and the output. Thus, a new training algorithm known as *resilient backpropagation* was examined.

The performance obtained when the training algorithm was changed was presented in **Figure 68**. The training algorithm was changed with the hope that the performance would improve.

```
>> x = (dinputs)';
t = (douts)';
net = feedforwardnet(10);
net.trainFcn = 'trainrp';
>> net = train(net,x,t);
view(net)
y = net(x);
perf = perform(net,y,t)

perf =

    3.2527
```

Figure 68. Output from resilient backpropagation training algorithm.

Despite the changed training algorithm, the performance error has not improved better. Apart from resilient backpropagation used other training algorithms such as variable learning rate backpropagation, one step secant, Polak-Powell Conjugate Gradient, Fletcher-Powell Conjugate gradient, Conjugate gradient with Powell and BFGS Quasi-Newton were all tried on the inputs and unfortunately, the error performance was not promising. Therefore, the available option to be explored before final conclusion could be made on feedforward network with respect to using it to predict the *mortality* is to remove the recurrence column to examine if that will assist the network to learn the relationship between the inputs and outputs.

Therefore, it means the number of input becomes 20 [311x20] while the output remains one [311x1]. The performance error is as shown in **Figure 69**.

```
>> x = (dinputs)';
t = (douts)';
net = feedforwardnet(10);
net.trainFcn = 'trainbfg';
net = train(net,x,t);
view(net)
y = net(x);
perf = perform(net,y,t)

perf =

    4.0746
```

Figure 69. The performance measurement of 20 inputs with a changed training algorithm.

The performance measurement shown in **Figure 69** above is a clear indication that changing the training algorithm and also the exclusion of recurrence column has not provided any added advantage to the network in terms of the performance. That is, the afore -mentioned efforts have not assisted in any way to enhance the effective learning of the relationship between the inputs and the outputs. Conclusively, based on this type of dataset, it can be aid deductively that feedforward neural network is efficient to predict the recurrence of mortality. However, the same type of network was not able to accurately predict the mortality (statuslatest) of patients using the same dataset. Thus, the need to examine another type of neural network becomes imperative. Hence, Elman Neural Network (ENN) as discussed in **Chapter 4** would be examined in **Section 5.6**.

5.6 Elman Neural Network for the prediction of Recurrence and Mortality

Having examined the feedforward network in **Section 5.5**, it was observed that the network was able to effectively predict the recurrence condition of the patient; however, predicting the status of the patient as far as mortality was concern was not effective. Therefore, the call on Elman Neural Network (ENN) cannot be over emphasized. Similar to the approach used for feedforward neural network, the network would be trained to make prediction of recurrence and after that; it would also be called to make prediction for mortality. In both cases, recurrence and statuslatest would be excluded from the inputs.

5.6.1 ENN for Recurrence prediction

Based on ENN methodology, the summary of the training is as presented in the **Figure 70**. It is important to note that the default training algorithm for ENN was *traingdx*, that is, Variable Learning Rate Backpropagation.

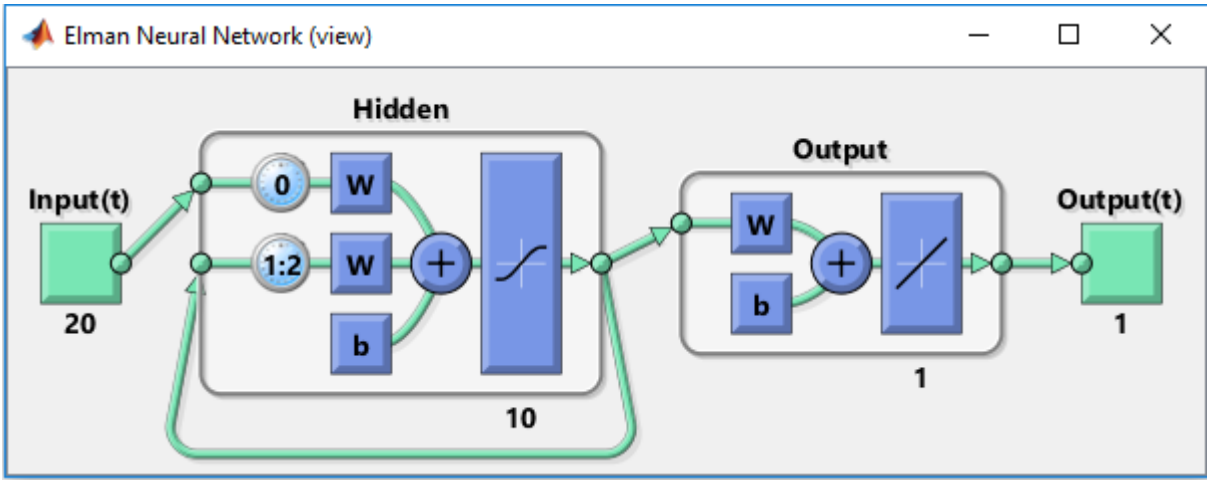


Figure 70. The training summary of ENN on the real data

However, the output produced based on this algorithm was not effective. That is, the network failed to learn the relationships between the inputs and the outputs. The default algorithm was later changed to Levenberg-Marquardt algorithm. The output from the default algorithm was presented in the **Figure 71**. In addition, other algorithms were also tried to see if the network can learn the relationship between the inputs but none was able to effectively learn the relationship.

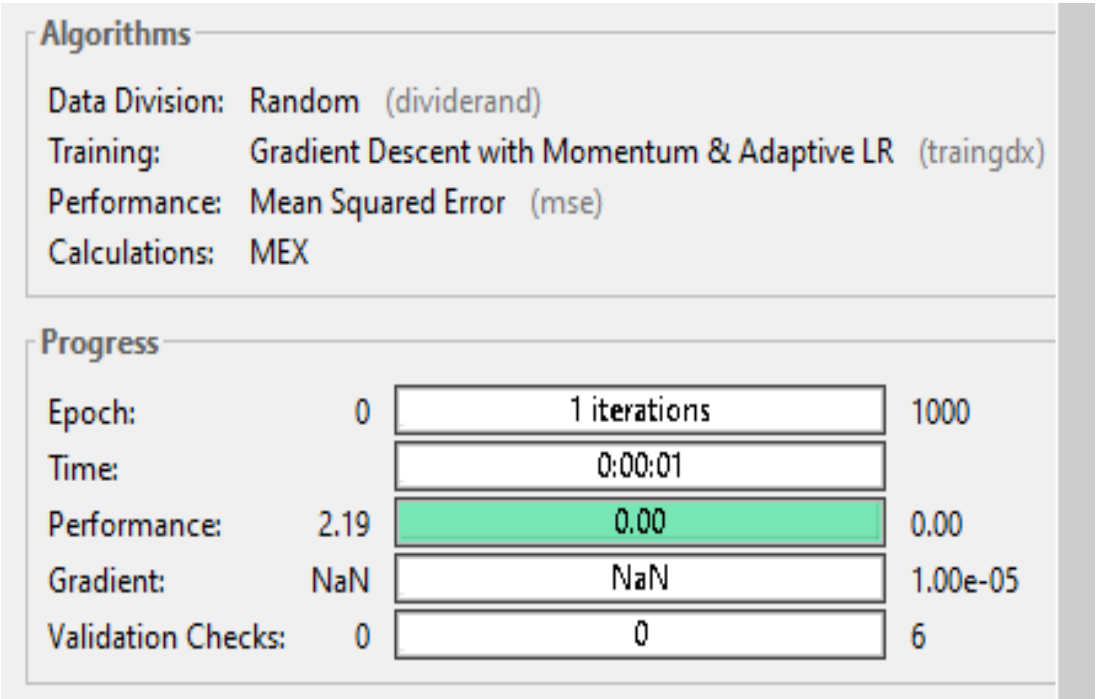


Figure 71. The default training algorithm for ENN.

From the output error performance produced in **Figure 71**, the network can't be trusted to produce a reasonable output. There are a lots of concerns based on Figure 71. The epoch, performance and gradient are enough pointers to indicate that ENN is not suitable for the dataset under consideration.

To further examine the effectiveness of ENN, the performance error was examined as shown in **Figure 72** which further indicates that no learning has taken place.

```
>> A = (dinputs)';
B = (douts)';
X = con2seq (A);
T = con2seq (B);
net = elmannet(1:2,10);
[Xs,Xi,Ai,Ts] = preparets(net,X,T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
Y = net(Xs,Xi,Ai);
perf = perform(net,Ts,Y)

perf =

    2.2870
```

Figure 72. Performance measure of ENN on the real data.

Therefore, based on the value of the expected output and neural output presented in the **Figure 73**, it is evident that the network has not learned the relationship. In that case, there's no sense to proceed further on using the network for predictions in the case of prediction of recurrence and mortality respectively.

Expected_Output	Neural_Output
[1]	[1.1936]
[0]	[1.4292]
[0]	[1.0123]
[1]	[1.6906]
[0]	[2.5031]
[1]	[1.9497]
[0]	[0.7234]
[0]	[1.5733]
[1]	[1.4787]
[0]	[1.4930]
[1]	[2.0015]

Figure 73. Expected and neural output of ENN on the real data.

The most appropriate thing to do in the advent of the afore-mentioned would be to try another neural network from the types neural networks presented in **Chapter 4**. Hence, Layer Recurrent Neural Network would be examined in the **Section 5.7**. However, it is important to quickly mention that the fact that ENN failed to learn the relationship between the inputs and output when applied to dataset in **Appendix IV** does not make it irrelevant. It was just an indication that ENN was not suitable for the dataset under consideration. Thus, a neural network can be effective on a particular data and might fail to perform similar effectiveness on another dataset with different variables. Hence, it is imperative to understand the dependencies of the variables within the dataset so as to be able to select an appropriate neural network that can be used in the analysis.

5.7 Layer Recurrent Neural Network (LRNN) for the prediction of Recurrence and Mortality

Feedforward and Elman neural networks have both been analysed with the aim of using both networks to predict recurrence and mortality of tongue cancer patients. However, the prediction performance in both cases was not satisfactory. Thus, the need to try another network was necessary. Layer recurrent neural network has been touted to provide a better performance than the duo of feedforward and Elman respectively.

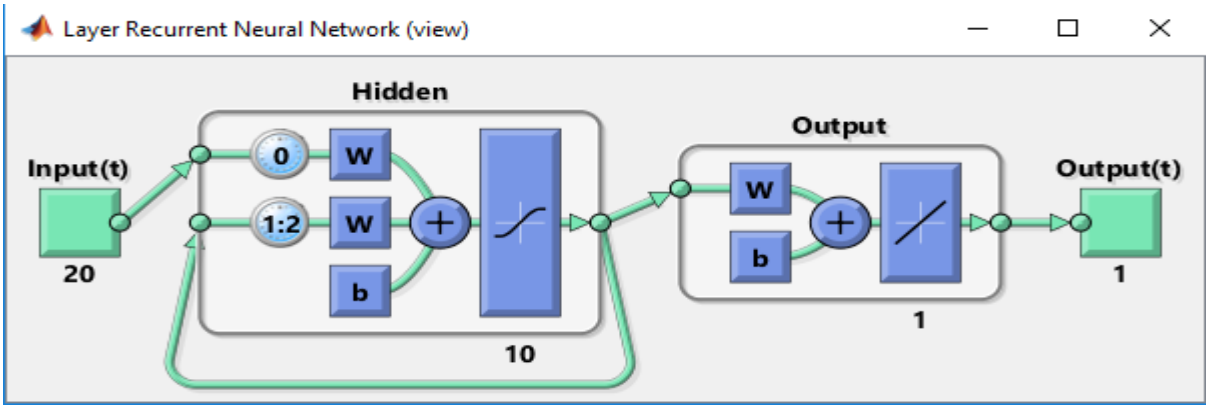


Figure 74. Layer recurrent neural network for prediction of recurrence and mortality.

As with earlier discussed neural networks of feedforward and Elman, 20 inputs are fed into the system [311x20]. Recurrence serves as the only output [311x1]. The summary statistics of the training process is presented in **Figure 74**.

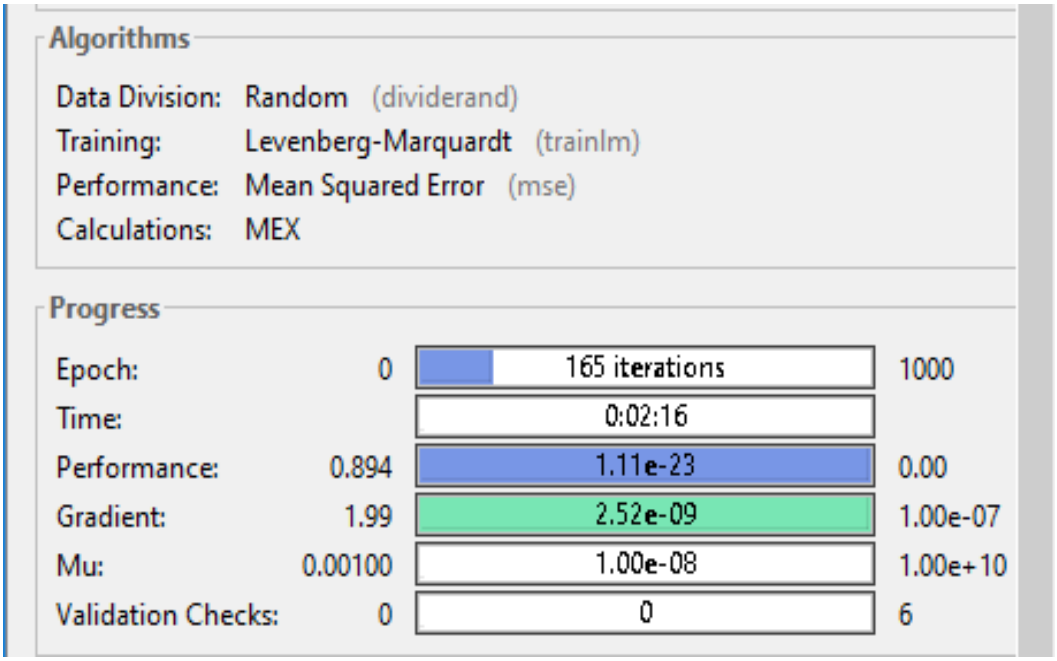


Figure 75. Training algorithm for later recurrent neural network

The training algorithm used was 'trainlm ' of Levenberg-Marquardt and it was effective based on some of the parameters shown in the **Figure 75**. The training converges at a reasonable time and also the number of iterations showed effective learning. The command code window for layer recurrent neural network is as presented in the **Figure 76**.

The error performance showed in **Figure76** from the training using Layer Recurrent appeared reasonable and that showed that the network had properly learned the relationships between the inputs and output.

Command Window

```
>> A = (dinputs)';
B = (douts)';
X = con2seq (A);
T = con2seq (B);
net = layrecnet(1:2,10);
[Xs,Xi,Ai,Ts] = preparets(net,X,T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
Y = net(Xs,Xi,Ai);
perf = perform(net,Y,Ts)

perf =

   1.1144e-23
```

Figure 76. The command window showing the performance of LRNN.

In terms of the expected and neural output, **Figure 77** gives a truncated output. Based on the performance error presented in **Figure 76**, it was evident that the network learned effectively.

Expected_Output	Neural_Output
[1]	[1.0000]
[0]	[2.3699e-12]
[0]	[-1.1555e-12]
[1]	[1.0000]
[0]	[4.4953e-13]
[1]	[1.0000]
[0]	[1.6349e-12]
[0]	[-4.6207e-13]
[1]	[1.0000]
[0]	[-5.3069e-14]

Figure 77. The expected and neural output for LRNN in recurrence prediction.

The extent of agreement between the expected and neural output could be further explored graphically as shown in the **Figure 78**.

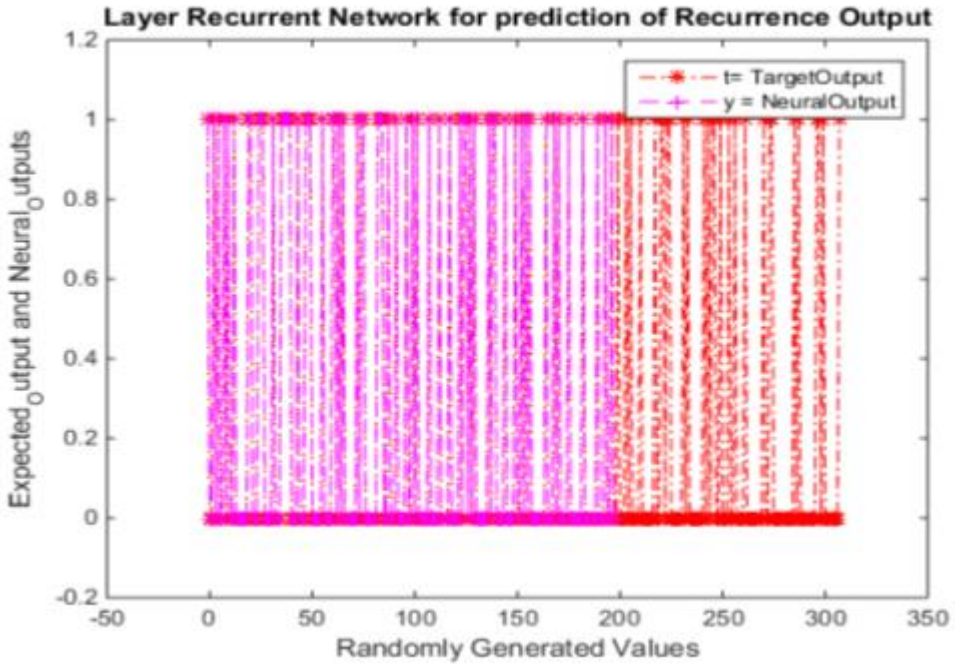


Figure 78. The expected and neural output for prediction of recurrence in layer recurrent neural network.

Similarly, the regression plot of parameters contained in each variable could assist to examine the concord between the expected and the neural output. Based on the plot presented in the **Figure 80**, it becomes imperative to conclude that the network had perfectly learned the relationship between the inputs variables and also the output. Thus, the learning was effective and that was evident as the neural output almost perfectly agreed with the expected output. Therefore, this gives a good leverage for the network to be used in prediction of recurrence based on the trained network. To finally examine if the network actually learned the inputs and output relationship, it is also very pertinent to examine the regression analysis.

After the training exercise using the training algorithm presented above, the regression plot is presented in the **Figure 79**.

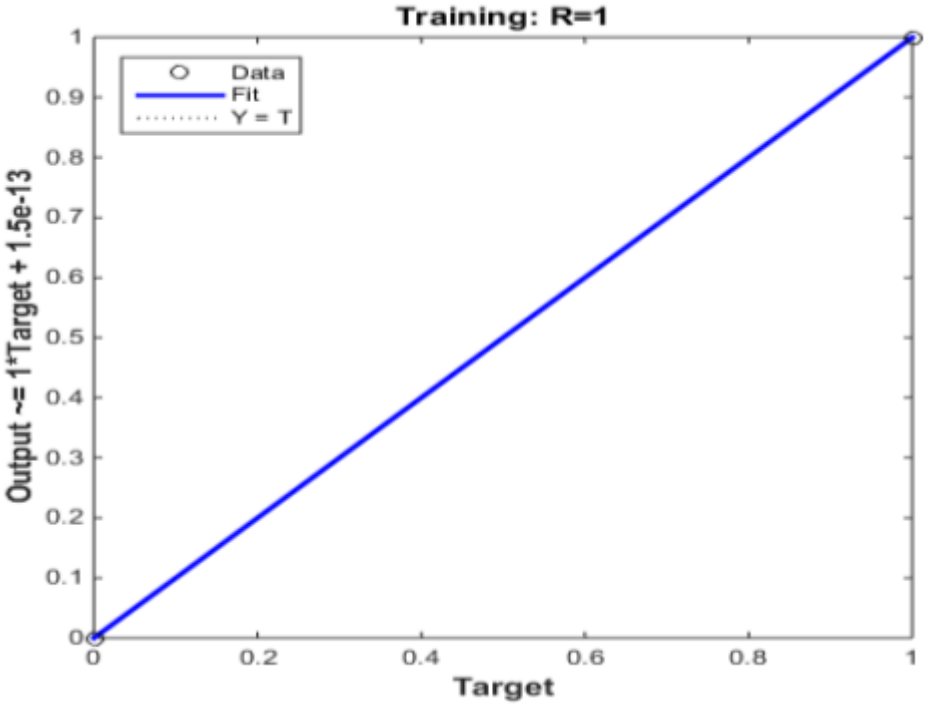
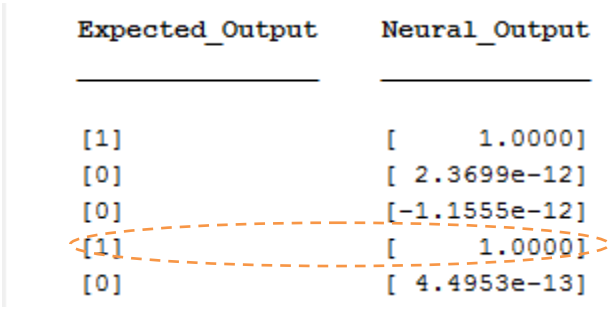


Figure 79. The regression plot of the training phase of the layer recurrent network for recurrence prediction.

As shown in **Figure 79**, the regression value was one (1). This means that the network effectively learned the relationship. Having established that the network had effectively learned, the next approach is to use the network to make prediction. This would be examined in **Section 5.7.1**

5.7.1 Prediction of Recurrence of Tongue cancer using Layer Recurrent Neural Network

The trained network is ready to be used for prediction of recurrence of tongue cancer. The trained network was fed with two inputs. One of the inputs was randomly selected from the given inputs to serves as a controlled prediction experiment. For the controlled prediction experiment, the third value was selected and the expected outputs are as given below:



Expected_Output	Neural_Output
[1]	[1.0000]
[0]	[2.3699e-12]
[0]	[-1.1555e-12]
[1]	[1.0000]
[0]	[4.4953e-13]

The expected inputs to be predicted was given as :

[2;2;7;1;1;6;2;12;72;2;2;2;1;6;3;1;1;0;0;0]

Given the prediction command in MATLAB, the prediction results is as shown in **Figure 82**.

```
>> pred_layerRecurrence_recurrence = net([[2;2;7;1;1;6;2;12;72;2;2;2;1;6;3;1;1;0;0;0]])

pred_layerRecurrence_recurrence =

    1.0000

fx >>
```

Figure 80. Prediction of recurrence using layer recurrent network.

As shown in above, the expected result to be predicted was 1. Interestingly, the network was able to predict that for that particular input, the expected recurrence would be 1. That is, for that particular patient with the same input as given above, the patient is expected to have tongue cancer recurrence as value 1 was obtained. The fact that the predicted value tallied with the expected value was enough to give the needed confidence that the network had actually learned the relationship between the inputs and the outputs. Therefore, the controlled prediction experiment was accepted. In the same way, the network would be used for an uncontrolled prediction exercise. In this case, an arbitrary input was formed as given below:

[1;2;6;1;1;6;2;10;70;2;2;2;1;5;3;1;1;0;0;1]

In the above input scenario, there was no idea of the expected output. All hope lies on the outcome of the prediction by the layer recurrent trained neural network. Therefore, based on the prediction command issued on MATLAB, the prediction is as given in **Figure 81**.

```
>> pred_layerRecurrenceNew_recurrence = net([[1;2;6;1;1;6;2;10;70;2;2;2;1;5;3;1;1;0;0;1]])

pred_layerRecurrenceNew_recurrence =

    1.1014

>>
```

Figure 81. Arbitrary input prediction for recurrence using layer recurrent network.

Based on the predicted output given in **Figure 81**, it can be said that for such given input, the patient would have recurrence of the tongue cancer. This is because the value obtained from the prediction was more than 1 and thus an indication of the possibility of recurrence.

Finally, having successfully predicted recurrence of tongue cancer using layer recurrent neural network, the next objective is to equally predict mortality using the same layer recurrent network.

5.7.2 Prediction of Mortality in Tongue cancer patients using Layer Recurrent Neural Network

The prediction of mortality have proven to be difficult using both feedforward and Elman neural networks respectively. In the case of feedforward neural network, the prediction of recurrence was effective while the prediction of mortality was very challenging. Prediction of both recurrence and mortality using Elman neural network was not effective. However, from the analysis presented in **Section 5.7.1**, it was evident that layer recurrent neural network perfectly predicted the recurrence of tongue cancer in the patients. Therefore, this section is aimed at using layer recurrence neural network to predict the mortality of the cancer. The performance measure of layer recurrent neural network is given in **Figure 82**.

```
Command Window
    1.1014

>> clear
>>
>> A = (dinputs)';
B = (douts)';
X = con2seq (A);
T = con2seq (B);
net = layrecnet(1:2,10);
[Xs,Xi,Ai,Ts] = preparets(net,X,T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
Y = net(Xs,Xi,Ai);
perf = perform(net,Y,Ts)

perf =

    0.4207
```

Figure 82. The performance output for the prediction of mortality.

The performance value obtained cannot be considered to be very good. Therefore, the output of the expected and the neural output are not in concord with each other. To address this issue, the first point of call was to change the training algorithm. The default training algorithm for layer recurrent was Levenberg-Marquardt. To address the above raised concern, the algorithm was changed both Variable Learning Rate Backpropagation and Resilient Backpropagation and there was no learning between the variables as shown in **Figure 83**.

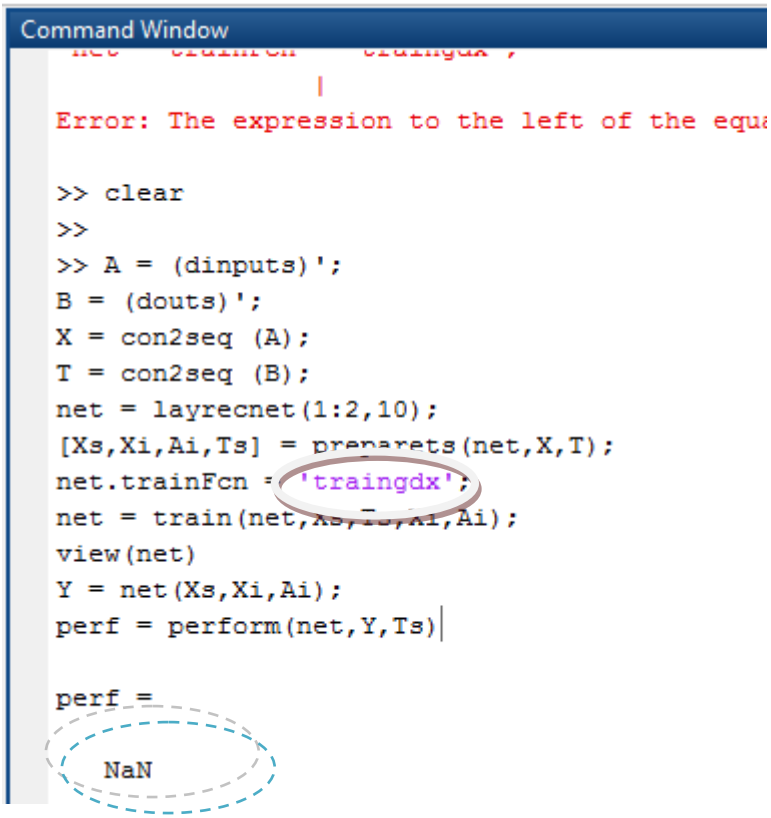


Figure 83. Output of the performance when the algorithm was changed.

Therefore, it became evident that changing the training algorithm would not provide a better error performance approximation. In addition changing to a new neural network such as time delay neural network (timedelaynet) might not represent a good approach because the dataset is independent on time. Similarly, nonlinear autoregressive neural network is also not considered to be a holistic approach because of the nonlinearity nature of the neural network.

It therefore becomes imperative to alter some of the internal design of the layer recurrent neural network. The number of the hidden neuron needs to be decreased or increased to examine if this will have any meaningful impact on the trained network. The default number of hidden neuron used in the network for the prediction of recurrence was 10 as shown in **Figure 75**. When the number of hidden neurons were decreased to 7, the performance error was not satisfactory. Consequently, the number of hidden neurons was later increased from the default value of 10 to 15 as shown in the **Figure 84**.

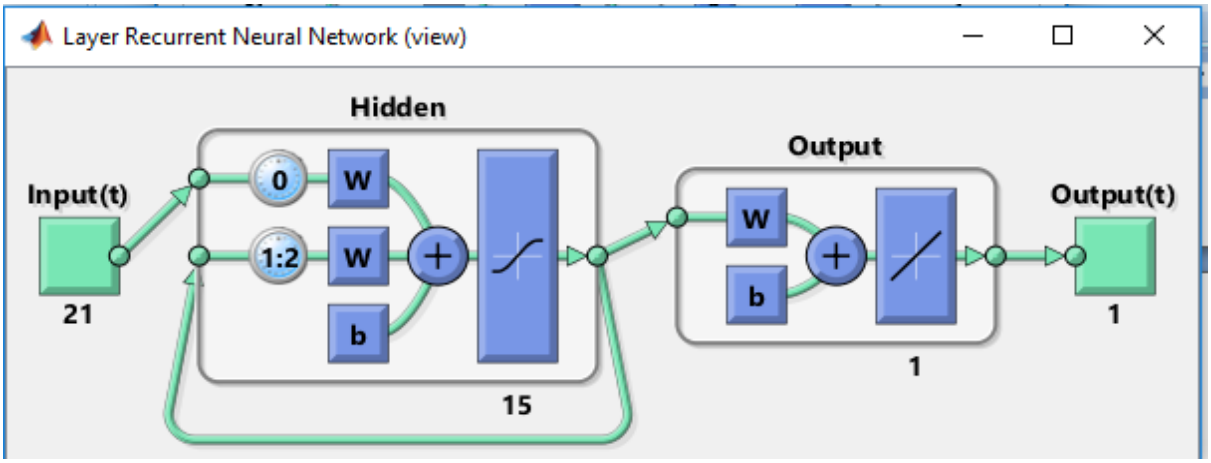


Figure 84. Layer recurrent neural network with increased number of neurons for mortality prediction.

The summary statistics of the trained network is also presented in **Figure 85**.

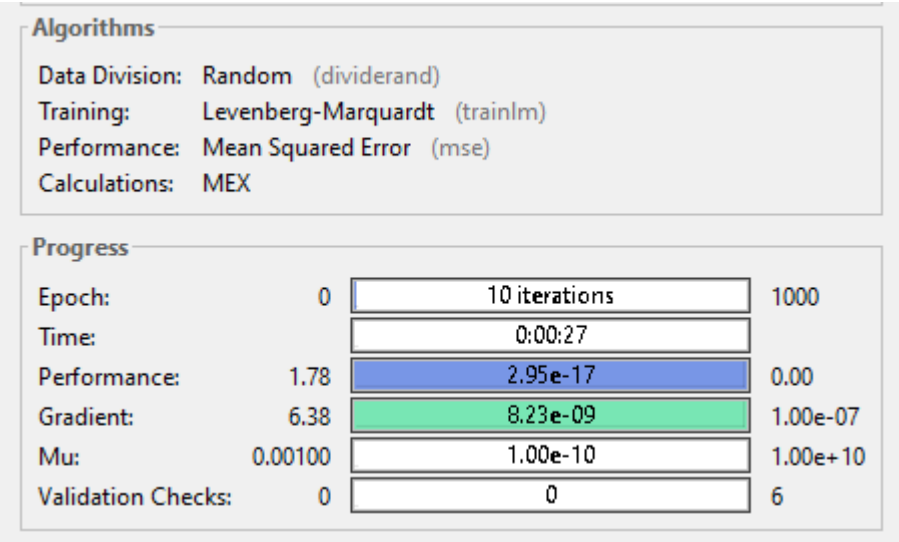


Figure 85. Training summary of layer recurrent with increased hidden neurons.

The default training algorithm was maintained and the performance error was quite reasonable to show that the level of agreement between the expected and neural output. The training performance showed a decreasing Mean Square Error (MSE) as seen in **Figure 86**. The low value of MSE indicated the network was well trained. Therefore, the target output is expected to be closer to the neural output.

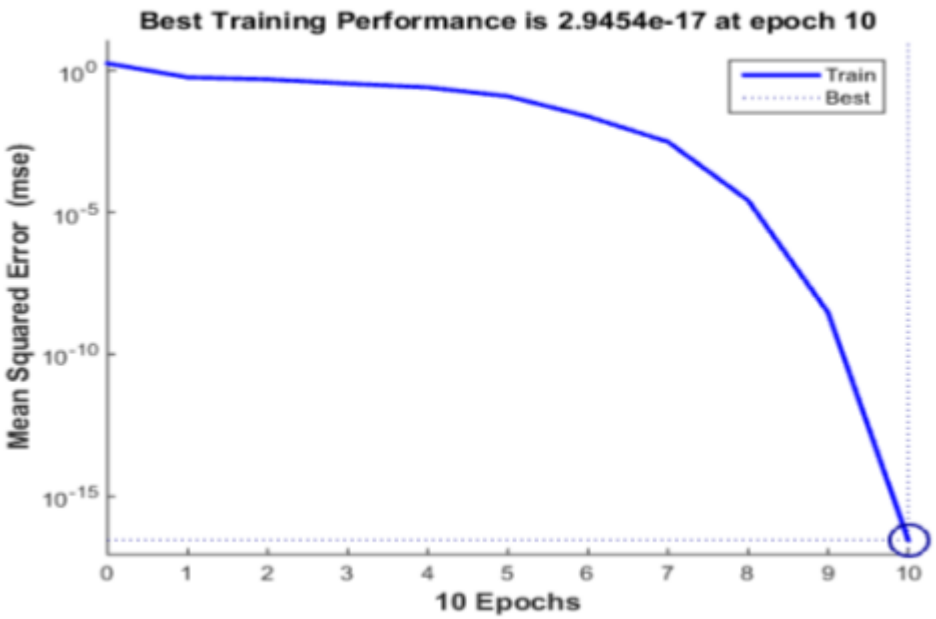


Figure 86. Mean Square Error performance of layer recurrent neural network.

The error is expected to reduce as the number of epoch increases. It is essential to give a truncated output of the values of the expected output and the trained output as given in **Figure 87**.

Expected_Output	Neural_Output
[1]	[1.0000]
[2]	[2.0000]
[0]	[1.2206e-09]
[0]	[7.8978e-10]
[2]	[2.0000]
[0]	[-6.1173e-10]

Figure 87. Expected and trained outputs after increased hidden neurons for layer recurrent.

Since the **Figure 87** above did not show all the expected and trained output, it is equally appropriate to have a glimpse of the relationship between these two important variables with the error histogram plot of **Figure 88**.

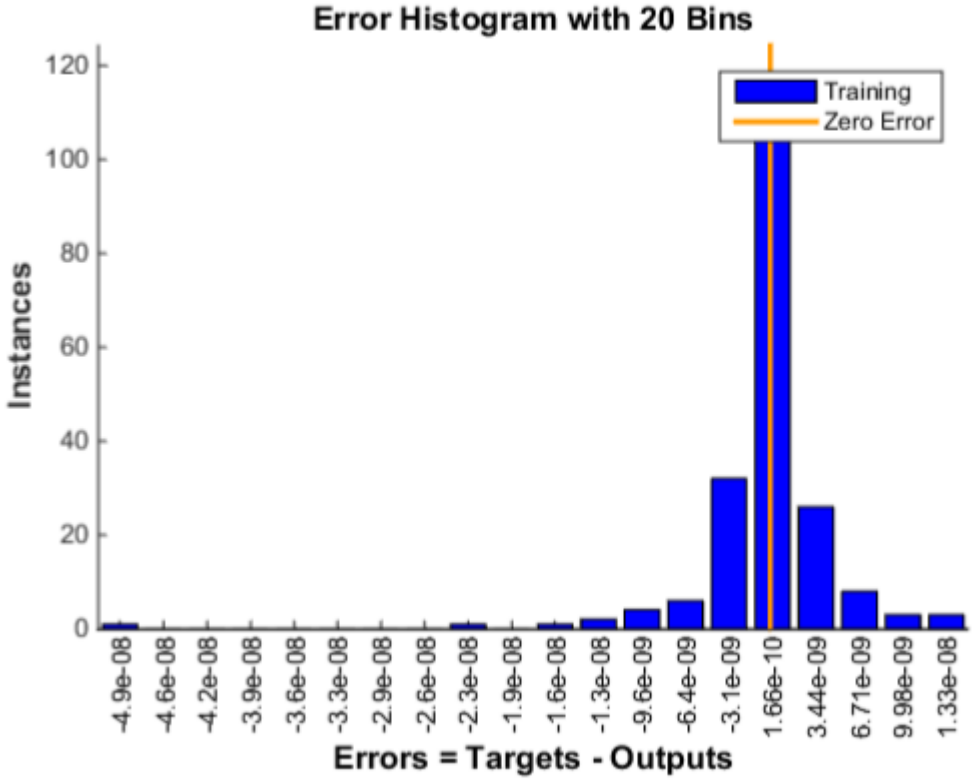


Figure 88. Error histogram plot of the difference between the expected and the trained value.

From **Figure 88**, it was evident that on a sizeable number of occasions that an error value was obtained. It means that the expected and neural outputs are the same on a number of occasions. With all the plots presented in this **sub-section 5.7.2**, it can be said without an iota of doubt that the network is ready to be used for the prediction of mortality because the network has been well trained and the difference between the expected output and the neural output is very insignificant. Therefore, **Section 5.8** will examine the use of this trained network to make predictions for mortality.

5.8 Prediction of Mortality using Layer Recurrent Neural Network

The trained network would be used to predict mortality as a result of cancer in a similar way that it was used to predict the recurrence of oral (tongue) cancer in the previous **sub sections 5.4.1 - 5.7** respectively. Controlled experiment prediction would be carried out with inputs with an expected mortality based on random selection of a row of the list of rows contained

in **Appendix IV**. This approach is poised to give further confidence on the performance of the network. In this case, row 4 was randomly selected. The input given in the 4th row was given as:

[2;2;0;0;1;6;2;20;77;2;1;1;1;20;0;0;1;0;0;0;0]

The expected prediction for the above input should be value 2 as shown below.

```
[1]                 [     1.0000]
[2]                 [     2.0000]
[0]                 [ 1.2206e-09]
[0]                 [ 7.8978e-10]
[2]                 [     2.0000]
[0]                 [-6.1173e-10]
```

From the trained network using prediction algorithm, it can be seen as shown in the **Figure 89**.

```
>> pred_layerRecurrenceNew_MORTALITY = net([[2;2;0;0;1;6;2;20;77;2;1;1;1;20;0;0;1;0;0;0;0]])

pred_layerRecurrenceNew_MORTALITY =

    1.6869

>>
```

Figure 89. The prediction of mortality using layer recurrent network.

Interestingly, the value obtained was 1.68 which can be approximated to 2 to indicate that the patient died of other causes. Conversely, a random input was formed and the prediction of this input using neural network was given below. The randomly generated input was

[1;1;0;0;0;7;2;20;70;2;1;1;1;20;0;0;1;0;1;0;0]

From the prediction algorithm in MATLAB, the prediction output is as given in **Figure 90.** It is

```
>> pred_layerRecurrenceNew_MORTALITY_random = net([[1;1;0;0;0;7;2;20;70;2;1;1;1;20;0;0;1;0;1;0;0]])

pred_layerRecurrenceNew_MORTALITY_random =

    0.8067

>>
```

Figure 90. Prediction of mortality using randomly predicted inputs.

The value of the predictions in both cases gave a decimal number. It means that results should be interpreted with caution. However, the performance error during the training phase for the network was perfect. It meant that the network perfectly learned the relationship. If that is the case, it means that the value can be round off without much concern as the network perfectly learned the relationship between the inputs and the output during the training phase. Therefore, from **Figure 90**, the value obtained was 0.8067 which means that the predicted value could be round off as 1 because the error performance of the trained network was insignificant. It can therefore be concluded that for the inputs above, the patient is likely to die of cancer than being alive.

The conclusion from various simulation exercises examined in this chapter, it can be said that the feedforward neural network can be used to predict recurrence of tongue cancer and not effective in the prediction of status of the patient. Elman on the other hand was neither effective for recurrence prediction nor for status prediction in oral tongue cancer patients. Layer recurrent neural network was the last neural network examined and it proved to be effective in the prediction of recurrence and mortality especially when the number of hidden neurons were increased in the case of prediction of the status of tongue cancer patients.

5.9 Analyses of the dependencies of variables contained in the dataset on the expected and neural outputs based on Layer Recurrent Neural Network

This section on the analyses of the dependencies of the variables on the expected and neural output is borne out of the failure of some of the examined networks in previous sections to perfectly predict the status of the patients. For instance, it was expected that the fact that the feedforward neural network effectively predicted recurrence of tongue cancer meant that it would also predict status of the patients. Similarly, much was expected from Elman neural network in terms of performance, but it failed to live up to the expectations. The only exception was the layer recurrent neural network as it was the only network examined that addressed the objectives of this thesis as highlighted in **Chapter 1**.

However it is important to mention that the performance of neural network is based on certain factors. The training algorithms, number of hidden neurons, nature of the dataset, the dependencies between the dataset to mention but a few are some of the factors that affect the performance of the network. Based on this fact, it would be erroneous to conclude that a certain neural network is the best. The best approach is to try various neural networks so as to examine which of the networks learned the relationship within the dataset to a reasonable extent. In the analyses of the dependencies among the variables, both recurrence of tongue cancer and status of the patients would be excluded from the inputs. That is, each of these two variables would serve as the output in the analyses. Two methods would be employed in the analyses.

First and foremost, recurrence of tongue cancer would be set as output while other variables with the exception of mortality (status) would constitute the inputs. That is, the network would have [311 x 20] and [311 x 1] as inputs and outputs respectively. In the second method, recurrence would be included as part of the inputs while statuslatest would be set as the output. In this case, the inputs would be [311 x 21] and output as [311 x 1]. Each of the variables would be varied to see the effect on the expected output when the particular variable in question has not been removed. To commence the analyses, the expected output is first given in **Figure 91**. This serves as the threshold or the significant level as the case may

be. Each variation output would also be presented and conclusion can be made based on the performance error

5.9.1 Analysis of variables dependencies for Recurrence prediction in oral Tongue cancer patients

The approach to the analyses has been explained in **section 5.9**. As explained previously, conclusion on the dependencies would be based on the value of performance error obtained.

```
>> clear
>>
>> A = (dinputs)';
B = (douts)';
X = con2seq (A);
T = con2seq (B);
net = layrecnet(1:2,10);
[Xs,Xi,Ai,Ts] = preparets(net,X,T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
Y = net(Xs,Xi,Ai);
perf = perform(net,Y,Ts)

perf =

   1.6944e-29
```

Figure 91. The network performance when all the inputs are used.

The first variable to be removed was *grade*, and when it was removed, the error performance was obviously reduced as given below

```
perf = perform(net,Y,Ts)

perf =

   2.7089e-25
```

When the value obtained from the removal of grade column was compared with the threshold as shown in the **Figure 91**, it could therefore be said that the variable called *grade* has no direct effect on the performance. This is because the error value obtained was almost the same. Thus, variable grade has no significant impact on the prediction outcome.

Similarly, variable called *grade_2* was removed and the performance was given as:

```
perf = perform(net,Y,Ts)

perf =

   2.8472e-20
```

An increased in the error means that the variable is an important variable. Though, it might not affect the prediction outcome greatly as the error was quite much. In the case of the removal of variable named budding, the effect on the network performance was quite insignificant as shown below.

```
perf = perform(net,Y,Ts)

perf =

   1.2801e-26
```

When the value obtained from the removal of budding was compared to the standard as given in the **Figure 91**, it was evident that the performance was very close and thus, it can be said that variable named budding has little or no effect on the performance. In the case of budding_2, the performance analysis is as shown below

```
perf = perform(net,Y,Ts)

perf =

   2.4055e-12
```

Budding_2 as a parameter has a meaningful impact on the output when compared to the performance presented in **Figure 91**. Thus, based on the variables examined so far, it can be said that the regrouped budding column known as *budding_2* has meaningful impacts on the performance of the system. Depth of invasion or simply depth as named in the variable column appeared to be a major parameter in the study of tongue cancer. Therefore, it is important to test its dependency on the performance of the network. When depth as variable was removed, the results obtained interestingly pointed to the fact the variable was very important in the prediction of recurrence of oral tongue cancer. The value obtained was given as

```
perf = perform(net,Y,Ts)

perf =

    0.1716
```

When it was removed, the performance went flat and the prediction gave a wrong result. The regrouped value for the column depth, that is, depth_2 showed that the variable was very important as the number of error increased as shown below.

```
perf = perform(net,Y,Ts)

perf =

   1.2197e-13
```

Therefore, *budding_2*, *depth and depth_2* have shown significant importance on the use of the network for prediction. Furthermore, *modified_stage* as a variable showed high level of significance as shown below.

```
perf = perform(net,Y,Ts)

perf =

    0.2036
```

That is, the *modified_stage* as a parameter was proven to be very important.

Similarly, time in month parameter had little or no effect on the performance when compared with the threshold.

```
perf = perform(net,Y,Ts)


perf =

   1.0674e-23
```

Based on the tested parameters so far, *budding_2, depth, depth_2, and modified_stage* had significant impact on the performance of the system. In addition, the age variable was examined and the result of the network performance was shown below.

```
perf = perform(net,Y,Ts)

perf =

   7.0699e-24
```

The result of the removal of age variable pointed to the fact that it has no significance on the usage of the network for prediction purpose. Also, when the age variable was regrouped to form another column, the performance was less significant on the performance of the network as shown below.

```
perf = perform(net,Y,Ts)

perf =

    1.4355e-18
```

Stage as one of the columns in the given dataset as always had been perceived to be one of the most important columns theoretically. It is therefore very important to test this importance on the performance of the network. Theoretically, the stage was considered to be one of the most important markers. It gives an insight into the present state of the cancer. When this marker was tested for significance using neural network, the performance obtained showed that it is also important. Though, it has no direct impact on the performance of the system as to be used for prediction but it was essential in other to have a meaningful prediction.

```
perf = perform(net,Y,Ts)

perf =

    4.1644e-15
```

In addition, age as a parameter has no significance on the performance of the system as shown below. The value obtained had little or no significance on the performance.

```
perf = perform(net,Y,Ts)

perf =

    4.1649e-25
```

Also for regrouped age known as *age_2*, was also subjected to dependency test and the result is as shown below.

```
perf = perform(net,Y,Ts)

perf =

   4.8548e-27
```

The result obtained was quite similar to the result obtained from the original age column. Hence, to briefly summarize the dependency test so far, it can be said that *budding_2, depth, depth_2, and modified_stage, and stage* are the columns that had impact on the performance with depth and budding_2 as very important columns and having the strongest impact on the prediction outcome. In the same vein, gender column was tested and the result was presented as below:

```
perf = perform(net,Y,Ts)

perf =

   3.0306e-19
```

From the above performance, it was evident that the gender column has no impact on the performance of the network. Similarly, the centre where the patients data was collected has little or no impact on the system for prediction purpose as shown below:

```
perf = perform(net,Y,Ts)

perf =

   1.3805e-21
```

The disease free month marker is as shown below. It is also one of the important markers theoretically. Therefore, it is pertinent to examine its importance as far as neural network was concerned.

```
perf = perform(net,Y,Ts)

perf =

   1.2560e-04
```

Based on the value obtained above, it became evident that it has a direct impact on the performance of the system. Hence, *budding_2, depth, depth_2, and modified_stage, stage and disease free months* are the important columns discovered so far.

The worst pattern of invasion was represented by WPOI column and the performance was given below.

```
perf = perform(net,Y,Ts)

perf =

   3.3464e-21
```

When WPOI was regrouped as WPOI_2, the column has little or no effect on the usage of the network for prediction purpose.

```
perf = perform(net,Y,Ts)

perf =

   6.5263e-20
```

For LHR and regrouped LHR_2, it can be said that the both value have significant effect on the performance of the system as shown in both values respectively.

```
perf = perform(net,Y,Ts)

perf =

   2.6547e-11
```

```
perf = perform(net,Y,Ts)

perf =

   1.1556e-15
```

Finally, for PNI and PNI_2, the outcome is as given below.

```
perf = perform(net,Y,Ts)

perf =

   2.1095e-25
```

```
perf = perform(net,Y,Ts)

perf =

   3.2313e-24
```

Based on the output, it becomes evident that duo of PNI and PNI_2 have no meaningful impact on the network performance. As stated in the introduction to this section, this section was aimed at identifying the most important markers that are needed to effectively predict the recurrence of tongue cancer in patients based on the data collected. Based on the dependency analyses presented in section 5.9.1, it can be said that *budding_2, depth, depth_2, and modified_stage, stage, disease free months, LHR and LHR_2* are the columns, parameter, variables or markers that have a direct effect on the network performance for prediction of recurrence. These important parameters are given and tabulated in **Table 8**.

However, it is imperative to mention that variable or column known as budding_2 cannot be obtained without the budding column. Hence, as a matter of inference and deduction, budding column becomes important marker because it would be needed to have the regroup budding_2 marker. Deductively, budding and budding_2 are considered to be a single marker. This is the same for depth and depth_2 and any other variable that has a regrouped extension.

Table 8. Important markers for the prediction of recurrence of tongue cancer.

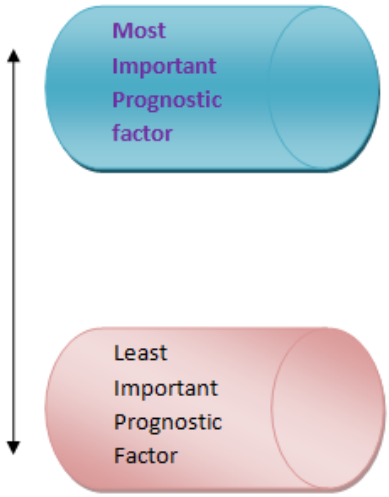
S/N	IMPORTANT MARKERS FOR RECURRENCE PREDICTION
1	Budding
2	Budding_2
3	Depth
4	Depth_2
5	Modified_stage
6	Disease free months
7	LHR
8	LHR_2
*	Stage. The value obtained for the stage as a prognostic factor served as the threshold for accepting or rejecting the parameter under test. Including it or not has no impact on the outcome. However, it was included in the subsequent analysis.

The markers presented in **Table 8** are enough to be used as input for layer recurrent neural network so as to have meaningful prediction of recurrence of tongue cancer. However, few other markers as deemed important may be added from the medical view point. The above markers presented in **Table 8** are based on neural network and they have been tested and proven to give correct prediction.

It is imperative to mention that budding and budding_2 are considered as a single prognostic factor. The same thing for depth and LHR. Therefore, the ANN identified four (4) prognostic factors and one other time dependent parameter-disease free month. These markers are then ranked to evaluate the order of importance. To do this, each of the interested marker are removed as input and the corresponding performance of the system was observed as presented in **Table 9**.

Table 9. Order of significance of the identified markers by ANN for recurrence prediction.

S/N (Position)	Order of Markers for Recurrence Prediction
1	Depth
2	Modified_stage
3	Budding
4	LHR



The results obtained from this showed that depth was the most important factor needed to effectively predict the recurrence of tongue cancer in patients. Similarly, LHR was ranked as the lowest prognostic factor from the identified prognostic factors identified by ANN.

5.9.2 Verification of the newly proposed dependent markers for the prediction of recurrence

This section is poised to validate the markers proposed above as enough markers to be used in prediction of recurrence of tongue cancer. The performance measure presented in **Figure 91** was computed using [311 x 20]. This was because it was the controlled or threshold performance measure. Thus, when the dependency analysis presented in **section 5.9.1**, the data iteratively changed to [311 x 19]. It was one variable less because there was one variable under consideration for dependency test. Based on Table 8, the variables under consideration for prediction of recurrence of tongue cancer in this section would be [311x 9] as shown in **Figure 92**.

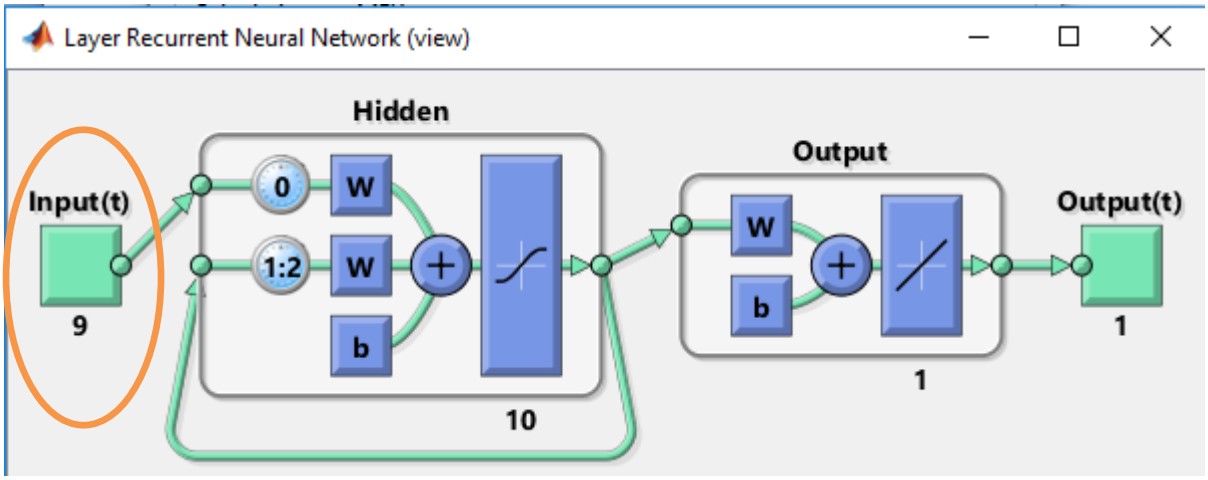


Figure 92. Training network for the new markers

Having reduced the number of variables or parameters from 20 variables to 9 variables, it is imperative to examine the network performance prior to the usage of the network for prediction. The performance of the network with 9 input variables is presented in **Figure 93**.

The main objective of this section is to demonstrate that the recurrence of tongue cancer can also be predicted with the newly proved markers presented in **Table 8**. In terms of the output, the network performance was shown in **Figure 93**. The network performance was best at 0.00012882.

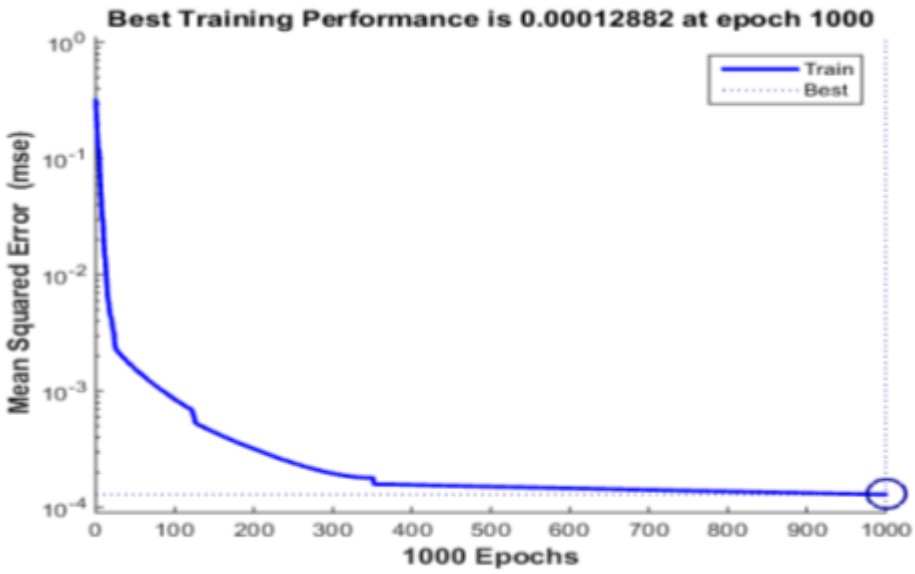


Figure 93. The performance of the neural network with the new inputs.

The truncated output presented in **Figure 94** showed the learning efficiency of the network.

Expected_Output	Neural_Output
[1]	[1.0130]
[0]	[0.0048]
[0]	[-0.0119]
[1]	[1.0142]
[0]	[-0.0216]
[1]	[1.0020]
[0]	[0.0011]
[0]	[0.0233]

Figure 94. Expected and neural output using the new markers.

It was easy to conclude that the network had learned the relationship inputs and output despite the fact that the network has been reduced drastically from 20 variables to 9 variables based on the dependency analyses given in **section 5.9**. Having established that the network had learned the relationship, the prediction exercise is given below. As demonstrated in the previous sections on prediction, the network will be used to make predictions. For instance, the network will be given the first row input to predict. The input was given as:

[9;1;1;8;2;1;1;3;1]

Since this is a controlled prediction, the value to be predicted is expected to be:

From MATLAB, the trained network was able to give prediction as:

```
>> pred_NewMarkers = net([[9;1;1;8;2;1;1;3;1]])

pred_NewMarkers =

    1.0450
```

Figure 95. The controlled prediction using the new markers.

The trained network was able to predict that for such input, the patient would have recurrence of tongue cancer. In the case prediction, it was very easy as the value of prediction lies between 0 and 1 as explained previously. In both cases, the values were more than 1 meaning that the patient is expected to have recurrence of tongue cancer.

The network was also used to predict randomly generated input. In this case, the input that was fed into the system was given below:

[7;1;1;7;2;1;1;1;1]

For the inputs above, the MATLAB prediction is as given in **Figure 96**.

```
>> pred_NewMarkers_random = net([[7;1;1;7;2;1;1;1;1]])

pred_NewMarkers_random =

    1.0125
```

Figure 96. Randomly generated input for the prediction of recurrence.

The trained network predicted that for the input above, the patient would have recurrence of tongue cancer. Therefore, it can be said the markers presented in **Table 8** are enough to make prediction of recurrence of tongue cancer in patients. That is, the other variables that are not included in the list of variables in **Table 8** are not needed to make the prediction of recurrence of tongue cancer. Lastly, the dependencies of the variables for the prediction of the status of the patients would be examined in section **5.9.3**.

5.9.3 Analyses of variables dependencies on the prediction of mortality of tongue cancer patients

One of the objectives of this thesis was to use neural network to predict the recurrence and mortality of tongue cancer patients. Having successfully analysed the dependencies of the variables on the performance of the system in the case of recurrence, the next approach was to use the same methodology to examine the dependencies of the variables on the prediction of mortality of tongue cancer patient. Recurrence of tongue cancer as a marker has a direct impact on the status of the patient. Thus, it is imperative to evaluate for the dependencies on status of the patient. However, it is imperative to mention that only the new markers presented in **Table 8** would be tested against the performance of the system in the case of status prediction. The network with all the inputs would have **[311 x 21]** with recurrence column included since it can now be correctly predicted as shown in the **Figure 97**. While mortality as the variable of interest serves as the output **[311 x 1].**

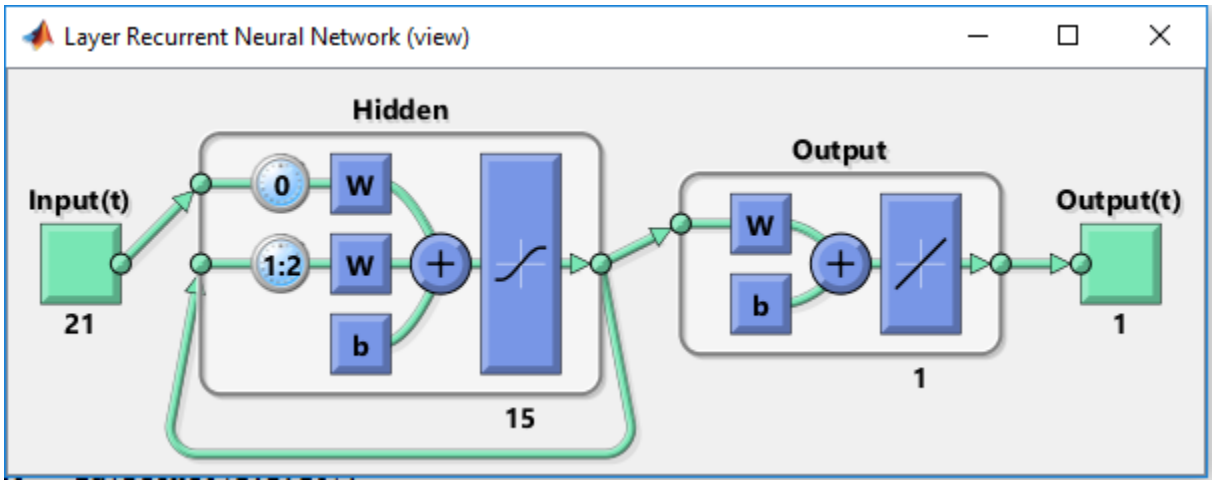


Figure 97. Input and output summary for the dependencies analysis.

Based on the above configuration, the network performance produced would serve as control in the analyses of the dependencies. In this case, only the variables that show sign of significance on the performance would be documented. The new marker that would be considered to have influence on the performance would be tabulated. Finally, the newly considered to be important markers would be used to make predictions for the mortality of the patients.

```
>>
>> A = (dinputs)';
B = (douts)';
X = con2seq (A);
T = con2seq (B);
net = layrecnet(1:2,15);
[Xs,Xi,Ai,Ts] = preparets(net,X,T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
Y = net(Xs,Xi,Ai);
perf = perform(net,Y,Ts)

perf =

   4.2454e-22
```

Figure 98. The performance of the network for prediction of mortality.

Only the variables that has significant effect on the performance of the network as presented in the **Figure 98** would be documented in this section. Therefore, it can be said from the dependencies analyses that variables *grade*, *budding_2 and depth* have no impact on the network performance. Grade_2, budding, depth_2 and modified_stage have shown importance on the performance of the system.

```
perf = perform(net,Y,Ts)

perf =

    0.4357
```

For instance the performance value of the network when modified_stage column was removed is as presented above. When the value obtained above is compared to the value given in the **Figure 98**, it became evident that it has significant impact on the performance. Time in months as a variable has been assumed by clinicians to have a direct impact on the status of the patient. Therefore, this assumption was put to test through the network. The performance value obtained was given as

```
perf = perform(net,Y,Ts)

perf =

    0.3995
```

From the performance value, it can be said that the variable time in months have a reasonable impact on the performance. Variables age, age_2, and stage have no importance on the performance of the system. Gender have been proven to be significant as shown below

```
perf = perform(net,Y,Ts)

perf =

    0.0680
```

Centre where the data was taken proved to be important as shown below:

```
perf = perform(net,Y,Ts)

perf =

   8.7755e-16
```

Similarly, another important parameter obtained was disease free months as presented below:

```
perf = perform(net,Y,Ts)

perf =

    0.6334
```

From the dependency analyses above, it can be said that the variables that have significant on the network are grade 2, budding, depth_2, modified_stage, time in months, gender, and disease free months. Worst pattern of invasion of invasion (WPOI) and regrouped WPOI known as WPOI_2, LHR, LHR_2, PNI and PNI_2 have little or no significant importance on the performance. Finally, the dependency of recurrence as an important parameter was examined and the result obtained was given as:

```
perf = perform(net,Y,Ts)

perf =

   8.1868e-23
```

Theoretically, recurrence was thought to have a direct influence on the prediction of mortality. When the value obtained above was compared with the value given in **Figure 98**, it became evident that these two variables, that is recurrence and mortality are independent variables. Thus, the examination of the dependency of each variable on mortality was justified. Based on the analysis of the variables presented in the section 5.9.3, the variables that were found to have significant impact on the prediction of mortality are presented in **Table 9**. These are grade 2, budding, depth_2, modified_stage, time in months, gender, disease free months and recurrence.

Sequel to the dependencies analyses presented in this section to predict mortality, the markers or variables found to have effect on the prediction capacity are given in **Table 10**. It is important to mention that time dependent factors such as time in months and disease free months are not considered as factors from the medical point of view and also from the medical researchers perspective. However, as a System Engineer working on neural network, these parameters are considered important by the network. Based the effect of these two parameters on the outcome, it would be therefore be worthwhile in the future for the clinicians, medical doctors and researchers in the medical field to examine time in months and disease free months as an important parameter.

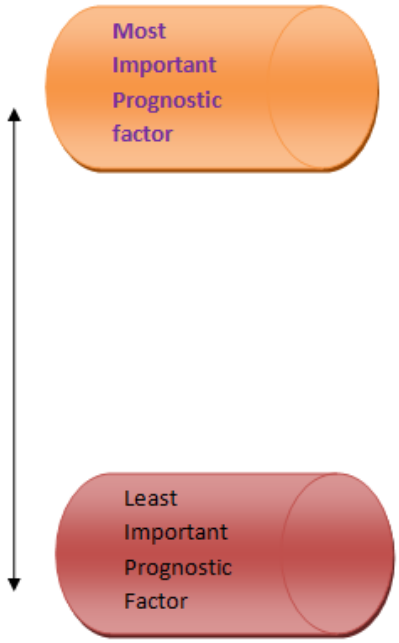
Table 10. Important markers for the prediction of mortality.

S/N	NEW MARKERS FOR MORTALITY PREDICTION
*	Grade and Depth
1	Grade_2
2	Budding
3	Depth_2
4	Modified_stage
5	Time in months
6	Gender
7	Disease free months
8	Recurrence
*	This means that both Grade and Depth are not marker highlighted by the neural network. However, the regrouped version are markers. Thus, they automatically becomes marker. Recurrence was also automatically added.

From **Table 10**, it can be concluded that, five (5) prognostic factors and also two (2) time dependent parameters (time in months and disease free months) were identified. Recurrence as a prognostic factor was automatically added because it was a target output like mortality. Having identified the necessary markers needed to effectively predict mortality, it is important to rank these prognostic factors in the order of importance of the mortality prediction capabilities. The results is as shown in **Table 11**.

Table 11. Order of significance of the identified markers by ANN for mortality prediction.

S/N (Position)	Markers for Prediction
1	Modified_stage
2	Depth
3	Budding
4	Grade
5	Gender



Modified stage was interestingly identified by ANN as the most important prognostic factor needed for mortality prediction. This further answers one of the objectives of this thesis. Hence, modified stage can be said to be a better prognostic factor. On the other hand, gender was ranked as the lowest in terms of the order of importance.

5.9.4 Using the new markers for the prediction of Mortality in tongue cancer patients

Having concluded that the variables presented in **Table 10** are enough to make predictions for the mortality in the case of tongue cancer patients, it is imperative to train these new variables presented in **Table 10** for prediction purpose. Prior to that, it is essential to examine how well the network was trained. This was presented in **Figure 99**.

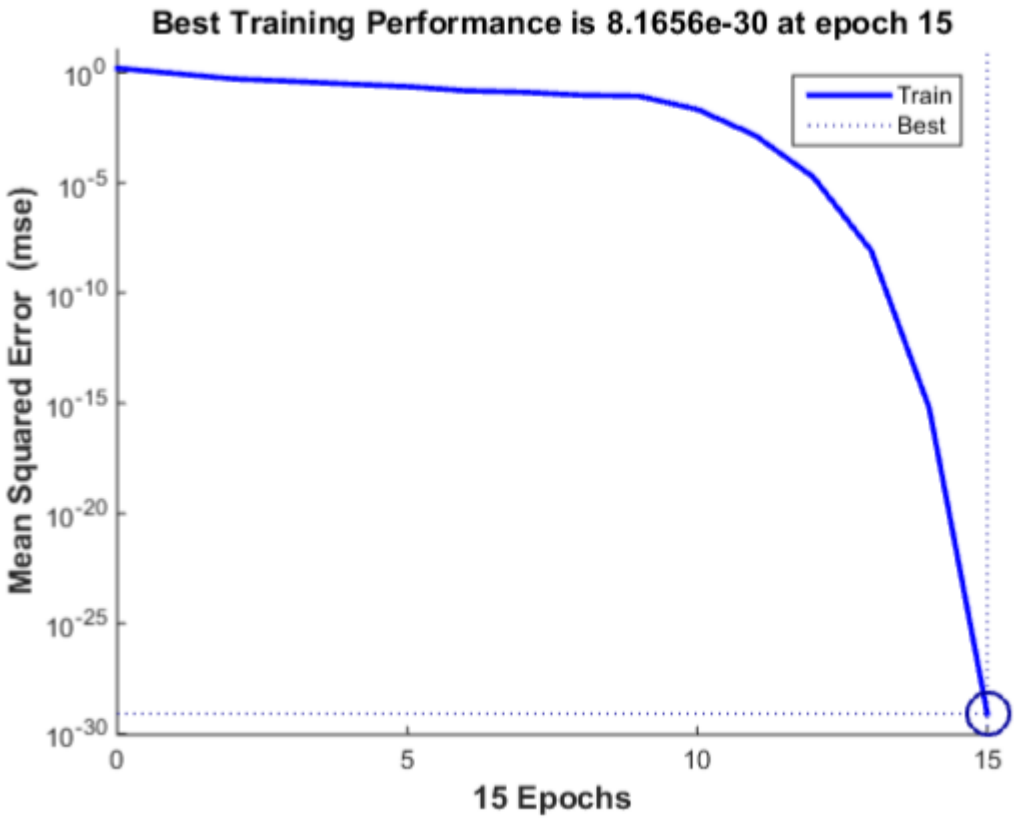


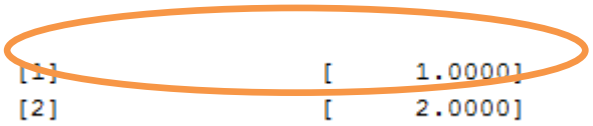
Figure 99. The training performance of the network with the new markers for mortality.

Based on **Figure 99** presented above, it was evident that the network had learned the relationship between the inputs and outputs perfectly. The network performance was far better than the performance presented in **Figure 98** where 21 inputs variables were used.

The fact that 7-9 inputs variables gave a better performance point to the fact that the markers that were suggested by the neural network were appropriate. Therefore, the network is ready to be used for prediction. Similar approach used in the previous prediction exercise would be done in this section. The network was used to predict the third input variable. The aim of this was to serve as a controlled experiment. This will give an insight to the level of acceptance or rejection of the prediction level as the case may be. Although, the markers suggested by the network are trusted but as an error prevention mechanism, it was appropriate to double check. The input presented to the network was:

[2;2;7;1;6;2;12;2;6]

The expected prediction would be given as



From MATLAB, the prediction was as shown in Figure 102.

```
>> pred_NewMarkers_MORTALITY = net([[2;2;7;1;6;2;12;2;6]])

pred_NewMarkers_MORTALITY =

    1.0000
```

Figure 100. Controlled prediction of mortality using the new markers.

With the prediction result shown in Figure 100, the patient under consideration with the set of the inputs above will die either as a result of cancer or other death. When randomly selected input of [2;2;4;1;10;2;5;1;5] was used, the output was shown in **Figure 101**.

```
>> pred_NewMarkers_MORTALITY_Random = net([[2;2;4;1;10;2;5;1;5]])

pred_NewMarkers_MORTALITY_Random =

    1.0665
```

Figure 101. Random prediction of inputs for mortality.

Based on the result given in **Figure 101**, the patient will not be alive. It can therefore be concluded from this chapter that ANN could be used to make predictions. In the same vein, it can be used to suggest important variables from the list of variables.

Table 12. Combined markers found to be important for ANN

S/N	Important Markers for both the prediction of Recurrence and Mortality
1	Grade
2	Depth
3	Budding
4	Modified_stage
5	Time in months
6	Gender
7	Disease free months
8	LHR
9	Recurrence

5.10 Sigmoid function on the output layer

The results presented in **Chapter 5** especially for the Layer Recurrent neural network was able to provide answers to the aims and objectives of the thesis as stated in the **Chapter 1**. This is because it was meaningful for engineers to generalize as far as numbers are concerned. That is, engineers generally round off numbers in the in the interpretation of results. This is not the same in medical parlance. Clinicians generally want a clear distinction between numbers. For instance, in engineering field, it is easy to classify the following numbers *2.345e-23* , *-5.34*, *-1.23* to all means zero. This is not the case in medicine. Clinicians and researchers needs a clear distinction between numbers and also the meaning of the numbers. Therefore, to eliminate confusion to the clinicians and researchers in medical fields who will later use this research as an assistive diagnostic system, sigmoid function has been touted to provide distinction of the predicted results. Sigmoid function would be used on the output layer to bound the expected result of the neural outputs. Sigmoid function is the same as activation function. It helps to find multiple applications to neural network (Chen and Cao 2012,2015; Anton et al 2017). Therefore, in this **sub-section 5.10**, sigmoid function would be applied using Layer Recurrent neural network to bound the neural output from the trained network.

The fact that sigmoid function is monotonic and real value has made it to be widely used in engineering field. There are various examples of sigmoid function but the nature of the output determines the type of sigmoid function to be used. In this thesis, recurrence as an output from the given dataset has output of 0 and 1, therefore, logistic function also known as logsig was used. The equation of some of the widely used activation functions was presented in **Section 2.2** of this thesis. In furtherance, the differentiable, non-negative first derivative feature of logsig was used to set the expected neural output to values between 0 and 1 by changing the default setting of the neural network to logsig as shown below:

```
net.layers{2}.transferFcn = 'logsig'
```

This step was necessary to eschew the possibility of negative answers or answers that are greater than 1.

5.10.1 Prediction of Recurrence of Tongue Cancer based on sigmoidal output function.

Based on the equation of logistic function presented in **section 2.2**, it mathematically means that when the neural output are zero or negative, the sigmoidal activation function would turn the neural to be 0.5 based on the equation. Similarly, when the neural output is 1 or greater than 1, the sigmoidal activation function would subsequently apply the formula and output will be produced in the region of 0.9-1 as shown in Table 13.

Table 13. Sigmoidal function analysis on the neural output

S/N	neuralOutput	SigmoidalTrained_Output	Meaning
1	Less than or equal to Zero (< = 0)	0.5	No Recurrence
2	Greater than or equal to zero (>=0)	0.9 - 1	Recurrence

Table 13 formed the basis for the interpretation of the prediction of recurrence. When the training of the network was carried out with a changed output layer setting, the following was obtained as shown in **Figure 102**.

Expected_Output	SigmoidalNeural_Output
[1]	[0.9996]
[0]	[0.5000]
[0]	[0.5000]
[1]	[0.9998]
[0]	[0.5000]
[1]	[0.9998]
[0]	[0.5000]
[0]	[0.5000]
[1]	[1.0000]
[0]	[0.5000]
[1]	[1.0000]
[0]	[0.5000]

Figure 102. Neural output with activation function.

With a brief output of the expected output and sigmoidal neural output shown in **Figure 102**, it can be said that the performance error was not bad. Similarly, **Figure 102** can best be understood based on the explanation given in **Table 13**. Based on this, the trained network can be used for prediction while considering the explanation presented in **Table 13**. In summary, the target output that corresponds to value zero (0) have been marked as 0.5 while the target output that corresponds to 1 have been marked with a value 0.9 to 1 by the activation function as shown in **Figure 102**.

Having shown the performance of the trained network with regards to the expected target and the sigmoidal neural output, this section was aimed to make prediction from the trained network. First and foremost, a random row was selected so as to serve as to provide some level of confidence on the performance of the network. To this end, the following input was selected to be used for prediction:

[2;2;0;0;0;3.5;1;60;72;2;1;2;6;60;1;1;3;1;0;0]

From the trained network, the prediction of the above input gave

```
>> pred_layerrecurrence_SigRecurrence = net ([[2;2;0;0;0;3.5;1;60;77;2;1;2;6;60;1;1;3;1;0;0]])

pred_layerrecurrence_SigRecurrence =

    0.5001

>>
```

Figure 103. Prediction of recurrence from the sigmoidal neural output.

According to the explanation given in **Table 13**, it can be said without any iota of doubt that for the input above, the patient will have no recurrence of tongue cancer. This is because the predicted value has a sigmoidal neural output of 0.5001 and this was equivalent to 0 as explained in **Table 13**.

Conversely, the predicted input value of [2;2;0;0;0;3.5;1;60;72;2;1;2;6;60;1;1;3;1;0;0] could be manipulated to generate another input. The newly generated input can also be used for prediction in the Layer Recurrence trained network.

The newly generated input is given as [1;2;1;1;0;2.9;0;78;52;1;1;2;6;40;1;1;3;1;0;1] as the recurrence prediction of tongue cancer was presented in **Figure 104**.

```
>> pred_layerrecurrence_SigRecurrence = net ([[1;2;1;1;0;2.9;0;78;52;1;1;2;6;40;1;1;3;1;0;1]])

pred_layerrecurrence_SigRecurrence =

    0.5000

fx >>
```

Figure 104. Arbitrary input prediction of recurrence in sigmoidal function layer recurrence network.

Therefore, from the result of the prediction shown in **Figure 104**, it can equally be said that the patience is likely to have no recurrence of tongue cancer. With the sigmoid function, it became easier to communicate in the language of the clinicians as explained in **Table 13**. With the result obtained in this section, that is, on the prediction of recurrence of tongue cancer using sigmoid function of logsig for interpretation of results, it becomes imperative to examine if such meaningful results would be obtained in the prediction of mortality using the same techniques. This will be presented in the next section, **section 5.10.2**.

5.10.2 Prediction of Mortality using sigmoidal function output layer

Sigmoid activation of logsig is similarly used in the prediction of mortality. This was necessary to be able to have meaningful interpretation of the predicted values. From the dataset given, it could be observed that mortality columns have values 0, 1, and 2 respectively. The meaning of these values was already explained **Table 7** of this write up. However, based on the activation function formula given in **section 2.2**, **Table 14** could be formed as shown.

Table 14. Sigmoidal neural output for mortality prediction.

S/N	neuralOutput	Sigmoidal_NeuralOutput	Mortality_Meaning
1	Less than or equal to 0 (<=0)	0.5 but less than 0.73	Alive
2	Greater than or equal to 1 but less than 2	0.73 but less than 0.88	Died of Tongue Cancer
3	Equal to 2	Greater than 0.88 (> 0.88)	Died of other causes

Table 14 formed the basis for the interpretation of the predicted result from the trained neural network. Prior to that, it is imperative to examine the extent to which the network had learned the relationship between the inputs and the target output. The output from the learning is shown in **Figure 104**. It is however imperative to mention that in the prediction of mortality presented in **sections 5.9.1 and 5.9.4** respectively, both cases of patients that died are classified as a single case. That is, both cases of death as a result of cancer and death as a result of other reasons are considered as death in the interpretation. This is based on the algorithm of neural network. Therefore, it calls for further research on this point in the future work. That is, distinction between death due to tongue cancer and death due to other causes.

With the use of activation function in this section, it is hoped that the distinction pointed out above could be actualized as explained mathematically in **Table 12**. Based on the result between the target output and the sigmoidal neural output as shown in **Figure 105**, it can be seen that the network had effectively learned the relationship. This indicated that the trained network is ready to be used for prediction. However, the reason for using logsig has not been justified as shown in some of the output shown in **Figure 105**

Expected_Output	Sigmoidal_Mortality_Neural_Output
[1]	[1.0000]
[2]	[1.0000]
[0]	[7.7013e-05]
[0]	[4.3220e-05]
[2]	[1.0000]
[0]	[1.4218e-04]
[2]	[1.0000]
[0]	[1.3497e-04]
[2]	[1.0000]
[2]	[1.0000]
[0]	[2.3601e-04]
[2]	[1.0000]
[2]	[1.0000]

Figure 105. Sigmoidal output for mortality

As shown in **Figure 105**, the trained output have basically given the value in the negative region and this was not the intended aim of using logsig. Although, the network was effectively trained, and it divided the outputs into the values of 0 - for patients that would be live and also 1- for patients that will die due to tongue cancer or die as a result if other causes. This was the same result presented in **Sections 5.9.1 and 5.9.4** respectively. The training algorithm and no of hidden neurons have been changed, but no success was achieved with these attempts. Therefore, it seems that changing the output layer to 'logsig' work best for output that has binary values. The *mortality (statuslatest)* column could be regrouped as *mortality_2* column which would contain binary values (alive [0] and die [1]) for future work using sigmoid function for easy interpretation of results.

6. CONCLUSION AND FUTURE WORK

Artificial neural network is an important tool used in information technology for analysis, classification, pattern recognition and prediction of outcomes. Therefore, its importance in medical field is also receiving attention in recent years. Despite this tremendous application of ANN, several factors have been known to affect the performance of the network. The training algorithm has been considered to be one of the most important factors that affect the neural network's performance. Similarly, nature of variables, linearity and non-linearity of the parameters, number of hidden neurons and preventing overfitting of data are some of other factors to be considered in other to have a neural network with effective performance. Therefore, it would be a fallacy to conclude that a neural network is the best. This is because a network might not work best on a dataset A and the same neural network would put up a magnificent performance when used on another dataset B. It means that all the known neural network types are useful in one capacity or the other. Although, it can be said that a neural network performs better than the other based on certain algorithm continued within the neural network.

Having said that, it is imperative to therefore examine the results obtained from the simulations presented in **Chapter 5** of this thesis. Based on the dataset analysed as contained in **Appendix IV** of this write up, it can be concluded that the feedforward neural network was able to predict the recurrence of tongue cancer in the patients. However, the feedforward network was not able to make predictions for the mortality (statuslatest) of the patients. As a result of this, the training algorithms, and number of hidden neurons were adjusted in anticipation of an improved performance but the performance measure was not improved. In that case, it was concluded that feedforward neural network can also predict recurrence and not mortality of the tongue cancer patients. Therefore, Elman neural network was considered.

Elman neural network could not effectively learn the relationship between the inputs and output of **Appendix IV**. The performance error obtained was exceedingly large. All the known training algorithms were used on the data but the performance was not improved. Therefore, Elman neural network was neither able to predict recurrence nor mortality of tongue cancer patients. This is because Elman neural network was not based on full dynamic derivative calculation. The failure of Elman neural network to have full dynamic derivative calculation led to the use of neural network that was based on full dynamic derivative calculation algorithm. Hence, layer recurrent neural network was touted as an option.

Layer recurrent neural network uses both *fpderiv and bttderiv*. Therefore, layer recurrent neural network was able to perfectly learn the relationship between the inputs and outputs and consequently, the prediction of recurrence of tongue cancer in the patients was successful. However, the prediction of mortality on the other hand was initially very challenging. The training algorithm was changed but no improvement on the performance. When the number of hidden neurons was increased, the network performed effectively and the prediction of mortality of the patient was possible. It is therefore imperative to mention that layer recurrent neural network was able to predict both the recurrence and mortality of tongue cancer patients.

Clinicians and researchers spend time, efforts, and money to gather variables for the purpose of analyses. Although, these variables are considered by these clinicians and researchers to be important in their analyses as far as their field of study is concerned. However, from the system's engineering point of view and data science and machine learning to be precise, it is pertinent to examine the variables for dependencies. This was aimed to reduce the stress and time put in by the researchers to gather these variables by suggesting the most important variables needed to have a meaningful prediction using artificial neural network. When the variables were analysed for dependencies, the number of inputs in the case of prediction of recurrence of tongue cancer in patients were reduced from 20 variables (11 prognostic factors) to 9 variables (5 prognostic factors) as shown in **Tables 8**. Moreover, the prediction of recurrence and mortality using these 9 variables gave meaningful and correct results.

Conversely, in the prediction of mortality, the dependencies of the variables examined in **Section 5.9.4** reduced the 21 variables (12 prognostic factors, that is, with the addition of recurrence) or markers that were contained in the dataset of **Appendix IV** to 8 variables (6 prognostic factors). These 8 variables were also used for the prediction of mortality or statuslatest of the patients and the results obtained was excellent with better error performance. These 8 variables were contained in **Table 10**.

The successful prediction of recurrence and mortality of tongue cancer patients further confirmed that ANN has an application in the medical field. This is because the results obtained would serve as a helping hand for the clinician to make an informed and futuristic decision about the patient. Moreover, reducing the number of variables to a more reduced variable and having the same efficient result was considered to be a meaningful application in the medical field. **Tables 8 and 10** were combined to give **Table 12**. Therefore, **Table 12** serves as the summary table that showed the needed markers for the prediction of recurrence and mortality in tongue cancer patients. It should however be noted that when using the markers presented in **Table 12** for the prediction of recurrence, recurrence itself as a prognostic factor should rather be the output. In both **Tables 8 and 10**, *modified_stage* as prognostic factor occurred in both tables. Hence, it can be said without an iota of doubt that *modified_stage* is a good prognostic factor in tongue cancer patients. This further answers one of the objectives of this thesis.

However, the differences in the research methodologies - approaches, presentation and interpretation of results in the Medical arena and Engineering field (Communication and Systems Engineering to be specific) has led to **section 5.10** where sigmoid function (activation function) was used. The aim was to present the results in a more understandable manner to the clinicians. However, the nature of the variables contained in the dataset did not help to actualize the intended reason why the activation function was used initially. Summarily, using activation function, the predicted values for recurrence was limited to 0.5 and 1 making it more readable. Where values between 0 and 0.5 meant that there was no recurrence and while values between 0.9 - 1 meant that was recurrence of tongue cancer based on the trained network prediction. Similar approach was used to limit the output layer for the prediction of mortality. However, this was not possible as the statuslatest column contained non binary values.

Based on the afore mentioned points, it can be said that using sigmoid function would not assist and thus, the results already presented for Layer Recurrent neural network in **Sections 5.9.1 and 5.9.4** would be explained to the clinicians as it gave better prediction performance.

Lastly, as pointed out in **Chapter 1** of this thesis, cancer is a dreadful disease and the need for proper classification in the prognostic approach becomes important. Based on the success recorded on the use of ANN on some parameters of the cancer patients in this thesis, other areas of concern in the cancer prognostication can be explored using ANN. This would serve as my recommendation for future work. From the discussion with my instructor, and from the simulation performed in this thesis, a lot of areas of research within the tongue cancer can be explored using ANN. Hence, for future work:

- ANN could be used to divide the patients into low risk and high risk. By low risk, it means that the patients would only need normal surgical approach while high risk means that the patients would need aggressive treatment like chemotherapy.

- Can sigmoid function be useful when the statuslatest column is regrouped to contain only alive and die cases (0 and 1 alone)?

- Can ANN be used when other prognostic factors are the main target outputs?

- The use of NARXNET (Nonlinear Autoregression with External Input) could be considered as a network to be used on the network. This is necessary especially when the clinicians come up with a new marker in the future that was not included in the dataset. Instead of training the network all over again, only the new marker is fed into the system as external inputs.

- The code could also be integrated into a website and it can be used by clinicians and doctors all over the world as presented in **Appendix V-VII.**

REFERENCES

Almangush A., Bello I., Keski-Säntti H., Mäkienen L.K., Kauppila H., Pukkila Matti, Hagström J., Laranne J., Tommola S., Niemien O., Soini Y., Kosma Veli-Matti., Koivunen P., Grenman R., Leivo I., and T. Salo (2013). *Depth of invsaion, tumour budding, and worst pattern of invasion: Prognostic indicators in early-stage oral tongue cancer.* Wiley Online Library. DOI 10.1002/hed.23380 . PP 811.

Almeida L.M. & T.B. Ludermir (2010). *A multi-objective memetic and hybrid methodology for optimizing the parameters and performance of artificial neural networks*. Neurocomputing vol 73 vol 7-9. PP 1438-1450.

Alwakeel M., & Z. Shaaban (2010). *Faze recognition based on haar wavellet transform and principal component analysis via Levenberg-Marquardt backpropagation neural network.* Eur.J.Sci. Res, vol 42.PP 25-31.

Anneroth G., Batsakis J., and Luna. M (1987). *Review of the literature and a recommended system of malignancy grading in oral squamous cell carcinoma*. Scand J Dent Res, vol 95.PP 229-249.

Anton Iliev, Nikolay Kyurikchiev and Svetoslav Markov (2017). *A family of Recurrence Generated Parametric Activation Functions with applications to Neural Network.* International Journal on Research Innovations In Engineering Science and Technology (IJRIEST), vol 2, Issue 1, January 2017.

Baran Tander, Atilla Ozmen & Ender Ozden (2015). *Neural Network Design for the Recurrence Prediction of Post-Operative Non-Metastatic Kidney Cancer Patients*. 9th International Conference on IEEE (ELECO), pp162-165.

Brandwein-Gensler M., Smith R.V., and Wang B (2010). *Validation of the histologic risk model in a new cohort of patients with head and neck squamous cell carcinoma*. Am J Surg Pathol vol 34: 676-688.

Bryne M, Koppang H.S., Lilleng R., Stene T, Bang G and Dabelsteen E. (1989). *New malignancy grading is a better prognostic indicator than boarders' grading in oral squamous cell carcinoma*. J Oral Pathol Med vol 18, pp 432-437.

Chen Z., and Cao F (2009). *The Approximation Operators with Sigmoidal Function*. Computers and Mathematics with Applications.vol 58, Issues 4. pp 758-765.

Chen Z., Cao F., & Hu J. (2015). *Approximation by Network Operators With Logistic Activation Function.* Applied Mathematics and Computations.vol 256, pp 565-571.

Delgrange V.N., Cabassud N., M. Cabassud., L.Durand-Bourlier and Laine J.M (1998). *Neural networks for prediction of ultrafiltration transmembrane pressure: application to drinking water production*. Journal of Membrane Science vol 150. PP 111-123.

Demuth H., Beale M., Hagan M. (2007). *Neural Network Toolbox 5*: Users Guide. Natick MA, The MathWorks Inc .

Elman L. Jeffery (2009). *Finding structure in time*. Cognitive Science vol 14 vol 2. PP 179-211.

Ferrari S., & R. Stengel (2005). *Smooth function approximation using neural networks*. IEEE Transaction on Neural Networks. Vol 16. PP 24--38.

Ferentinos K.P (2005). *Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithm*. Neural Networks Off. Journal of International Neural Network Soc., vol 18, no 7. PP 934-950.

Furundzic M., Djordjevic & A. Jovicevic Bekic (1998). *Neural networks approach to early breast cancer detection*. Journal Syst. Arch Vol 44. PP 617-633.

Ganly I., S. Patel., & J. Shah (2012). *Early stage squamous cell cancer of the oral tongue-clinicopathologic features affecting outcome*. Cancer vol 118. PP 101-111.

Goller C., & Kuchler. A (1996). Learning Task-Dependent Distributed Representations by Backpropagation Through Structure. IEEE International Conference on neural networks, vol 1, pp 347-352.

Hagan M.T., Demuth H.B., & Beale M.H (1996). *Neural Network Design*. PWS Pub Co. Boston, MA, USA. PP 3632.

Hagan M.T., & Menhaj M.B (1994). *Training feed forward networks with the Marquardt algorithm*. IEEE Trans. Neural Network Design, vol 5. pp 989-993.

Hammer B., Micheli A., Sperduti A., & Strickert Marc (2004). A general framework for unsupervised processing of structured data. Neurocomputing vol 57, pp 3-35.

Hammer B., Micheli A., Sperduti A., & Strickert Marc (2004). Recursive self-organising networks models. Neural Networks vol 17, pp 1061-1085.

Hassan Jouni, Mariam Issa, Adnan Harb, Gilles Jacquemod and Yves Leduc (2016). *Neural Network Architecture for Breast Cancer Detection and Classification*. IEEE International Multidisciplinary Conference on Engineering Technology (IMCET).

Heimes F., & B. van Heuveln (1998). *The normalized radial basis function neural network*. IEEE International Conference on Systems, Man, and Cybernetics. PP 1609-1614.

Ho C.M., K.H. Lam., W.I. Wei., S.K. Lau., and L.K. Lam (1992). *Occult lymph node metastasis in small oral tongue cancers*. Head Neck vol 14 PP 359-363.

Hopfield J.J (1982). Neural Networks and physical systems with emergent collective computational abilities. *In proceedings of the National Academy of Sciences of the USA*, vol 79 no 8, pp 2554-2558.

Hongying Meng, Nadia Bianchi-Berthouze, Yangdong Deng, Jinkuang Cheng and John P. Cosmas (2016). *Time-delay neural network for continuous emotional dimension prediction from facial expression sequences*. In IEEE Transaction on cybernetics vol. 46, no 4, pp 916-929.

Hu Y., Ashenayi K., Veltri, R., O'Dowd G., Miller G., and Hurst R. Bonner (1994). *A comparison of neural network and fuzzy c-means methods in bladder cancer cell classification*. In Proceedings of IEEE World Congress on Computational Intelligence; 6;p.3461-3466.

Huang C.L. & Wang C.J (2006). *A Genetic Algorithm (GA) based feature and parameters optimization for support vector machines*. Expert Syst. Appl., vol 31, no 2. PP 231-240.

Ibrahim M., Jemei S., Wimmer G., & D.Hissel (2016). *Nonlinear autoregressive neural network in an energy management strategy for battery/ultra-capacitor hybrid electrial vehicles*. Electric Power Syst. Res vol 136, pp 262-269.

Jakobsson P.A., Eneroth C.M., Kilander D., Moberger G., and Martensson. B. (1973). *Histologic classification and grading of malignancy in carcinoma of the larynx*. Acta Radio Ther Phys Bio vol 12, pp 1-8.

Jayadeva A., & S. Chandra (2002). *Algorithm for building a neural network for function approximation*. IEEE Proceedings- Circuits, Devices and Systems. PP 301-3017.

Kannathal N., Sadasivan K.A., & Lim Choo Min (2006). Elman neural networks for dynamic modelling of epileptic EEG. Published in the Engineering in Medicine and Biology Society. *In Proceedings of the 28th IEEE EMBS Annual International Conference*, pp 6145-6148.

Karabatak M & M.C. Ince (2009). An expert system for detection of breast cancer based on association rules and neural network. Expert Syst. Appl., vol 36, no 2. PP 3465-3469.

Keski-Säntti H., Atula T., Tikka J., Hollmen J., Mäkitie A.A., & Leivo I. (2007). Predictive value of histopathologic parameters in early squamous cell carcinoma of oral tongue. Oral Oncol vol 43, pp 1007-1013.

Kellermann M.G., L.M. Sobral, and S.D. da Silva (2007). *Myofibroblasts in the stroma of oral squamous cell carcinoma are associated with poor prognosis*. Histopathology Vol 51. PP 849-853.

Konstantina K., Themis P.E., Konstantina P.E., Michalis V.K & I.F Dimitrios (2015). *Machine learning applications in cancer prognosis and predictions*. Computational and Structural Biotechnology Journal vol, pp 8-17.

Levenberg Kenneth (1944). A Method for the Solution of Certain Non-Linear Problems in Least Squares. Quaterly of Applied Mathematics, vol 2. pp 164-168.

Lopez M. (2012). Application of SOM neural networks to short-term load forecasting: The Spanish electricity market case study. Electric Power Syst. Res. vol 91, pp 18-27.

Luiz Gonzaga Baca Ruiz, Manuel Pegalajar Cuellar, Miguel Delgado Calvo-Flores and Maria Del Carmen Pegalajar Jimenez (2016). *An application of non-linear autoregressive neural networks to predict energy consumption in buildings*. Energy Time Series Forecasting. Energies 9(9) PP 684.

Marquardt D.W (1963). *An algorithm for least-squares estimation of nonlinear parameters*. J. Society Ind. Appl.Math vol 11.PP 431-441.

Martinez-Gimeno C., Rodriguez E.M., Vila C.N., and Varela C.L (1995). Squamous cell carcinoma of the oral cavity: a clinicopathologic scoring system for evaluating risk of cervical lymph node metastasis. Laryngoscope vol. 105 (7 Pt 1), pp 728-733.

Marsh D., Suchak K., Moutassim K.A et al (2011). Stromal features are predictive of disease mortality in oral cancer patients. J Pathol vol. 223, pp 470-481.

MATLAB R2014b (2014). *Train and apply multilayer neural network and MATLAB toolbox*. MathWorks Inc 2014.

Nyanteh Y:D., Srivastava S.K., Edrington C.S., and D.A Cartes (2013). *Application of artificial intelligence to stator winding fault diagnosis in Permanent Magnet Synchronous Machines*. Electric Power Syst. Res. vol. 103, pp 201-213.

Oliver Gevaert & Bart De Moor (2009). *Prediction of cancer outcome using DNA microarray technology: past, present, and future*. Expert Opinion-Med. Diagn vol. 3(2). PP 157-165.

Po Wing Yuen A., Lam K.Y, Lam L.K., et al (2002). *Prognostic factors of clinically stage I and II oral tongue carcinoma- a comparative study of stage, thickness, shape, growth pattern, invasive front malignancy grading, Martinez-Gimeno score, and pathologic features*. Head Neck vol. 24, pp 513-520.

Paliwal M., & U. A Kumar (2009). *Neural networks and statistical techniques: A review of applications*. Expert Syst. Appl.,vol. 36,no 1. PP 2-17.

Rui Xu, Xindi Cai, Donald C., and Wunsch II (2005). Gene Expression Data for DLBCL Caner Survival Prediction with a combination of Machine Learning Technologies. In proceedings of the IEEE International Conference on Medicine and Biology. PP 894-897.

Safavieh E., Andalib S., & A. Andalib (2007). *Forecasting the unknown dynamics in NN3 database using a nonlinear autoregressive recurrent neural network.* In the Proceedings of the IEEE International Joint Conference on Neural Networks, Tehran Iran.

Setino R, & A. Gaweda (2000). *Neural network pruning for function approximation*. International Joint Conference on Neural Networks (IJCNN), Proceedings of the IEEE-INNS-ENNS. PP 443.

Setino R, & A. Liu (1997). *Neural network future selector*. IEEE Transaction Neural Network vol. 8, no 3, pp 654-662.

Shikha Agrawal and Jitendra Agrawal (1995). *Neural network Techniques for Cancer Prediction : A Survey*. The 19th International Conference on knowledge Based and

Intelligent Information and Engineering Systems. Elsevier: Available on Science Direct. Precedia Computer Science vol. 60. PP 769-774.

Silveira E.J., Godoy G.P., and Lins R.D et al (2007). Correlation of clinical, histological, and cytokertin profiles of squamous cell carcinoma of the oral tongue with prognosis. Int J Surg Pathol vol. 15, pp 376-383.

Spelt L., Nilsson J., Andersson R., & B. Andersson (2013). Artificial neural networks- A method for prediction of survival following liver resection for colorectal cancer metastases. Available on SciVerse-Science Direct. Elsevier: EJSO vol. 39, pp 648-654.

Sung-Bae Cho and Hong-Hee Won (2007). *Cancer classification using ensemble of neural networks with multiple significant gene subsets.* ApplIntell vol. 26, pp 243-250.

Tehmasebi Pejman; & Hezarkhani Ardeshir (2011). Application of Modular Feedforward Neural Network for Grade Estimation. Natural Resources Research vol. 20(1): 25-32.

Turkson R.F., Yan F., & Ahmed M.K. & Hu J. (2016). Artificial neural network applications in the calibration of spark-ignition. Elsevier.

Ubeyli E.D. (2007). *Implementing automated diagnostic systems for breast cancer detection.* Expert Syst. Appl.,vol. 33, no 4. PP 1054-1062.

Waibel A. (1989). Modular construction of time-delay neural networks for speech recognition. Neural Computation, vol. 1, no 1, pp 39-46.

Waibel A., Hanazawa T., Hinton G., Shikano K., & Lang K. (1989). Phoneme recognition using time-delay neural networks. In IEEE transactions on Acoustics Speech, and Signal processing, vol. 37, no 3, pp. 328-339.

Walczak S & N. Cerpa (1999). *Heuristic principles for the design of artificial neural networks*. Inf Softw.Technol., vol 41. PP 107-117.

Wang Y., Makedon F.S., Ford J.C., and J. Pearlman. Hyk Gene: *A hybrid approach for selecting marker genes for phenotype classification using microarray gene expression data*. Bioinformatics vol 15 pp 1530-1537.

Weijers M., Snow G.B., Bezemer P.D., & van der Waal (2009). *Malignancy grading is no better than conventional histopathological grading in small squamous cell of tongue and floor of mouth: retrospective study in 128 patients*. J.Oral Pathol Med vol 38, pp 343-347.

Yao X (1999). *Evolving artificial neural networks*. Proceedings IEEE vol 87 no 9. PP 1423-1447.

Ying Z., G. Jun, & Y. Xuezhi (2005). "A survey of neural network ensembles". *In: International Conference on Neural Networks and Brain (ICNN&B)*. PP 438-442.

Yuchun Tang, Yan-Qing Zhang, Zhen Huang, Xiaohua Hu, and Yichuan Zhao (2008). *Recursive Fuzzy Granulation for Gene Subsets Extraction and Cancer Classifications*. IEEE Transactions on Information Technology in Biomedicine; 2(6):pp 723-730.

APEENDIX I

	A r Number	B y Number
1	r	y
2	81.4724	4.3258
3	90.5792	4.2710
4	12.6987	2.0329
5	91.3376	4.0371
6	63.2359	3.4249
7	9.7540	3.0913
8	27.8498	4.2529
9	54.6882	3.4625
10	95.7507	5.2808
11	96.4889	4.6147
12	15.7613	3.5508
13	97.0593	3.6029
14	95.7167	5.2012
15	48.5376	3.7850

----------Output Missing----------

APPENDIX II

1	x1	x2	x3	x4	out
2	0.7136	5	-0.5517	-1.9366	-0.8014
3	0.4175	4	0.8309	3.8624	-5.0418
4	0.4342	7	-0.5706	-2.4305	3.4762
5	0.0526	1	0.5374	-3.7681	-21.7538
6	0.6555	8	-0.3840	-1.6456	3.8645
7	0.1077	5	0.8106	-3.0429	-6.9113
8	0.4975	4	-0.7135	-4.9736	-10.6668
9	0.7294	7	0.0564	1.4874	4.7497
10	0.1308	1	0.9988	-4.1920	-15.9633
11	0.3926	3	0.9677	4.9560	-11.2684
12	0.7184	8	0.8721	3.8993	3.0855
13	0.3281	4	0.6103	-2.4374	-1.9639
14	0.7202	1	-0.3664	3.9146	-15.9083
15	0.5870	7	-0.7169	2.1452	-5.6676
16	0.5065	6	-0.5508	-2.8987	-10.0151
17	0.7696	6	-0.4319	3.3364	-9.7302
18	0.5678	8	-0.1294	0.1139	3.1961
19	0.1117	4	-0.4174	2.5293	-15.1480
20	0.7049	7	-0.6500	0.3183	-7.0526
21	0.7932	5	-0.4365	4.2347	-16.4780
22	0.8056	8	0.0416	4.6472	-15.9915
23	0.9379	8	0.7710	-1.3231	3.2688

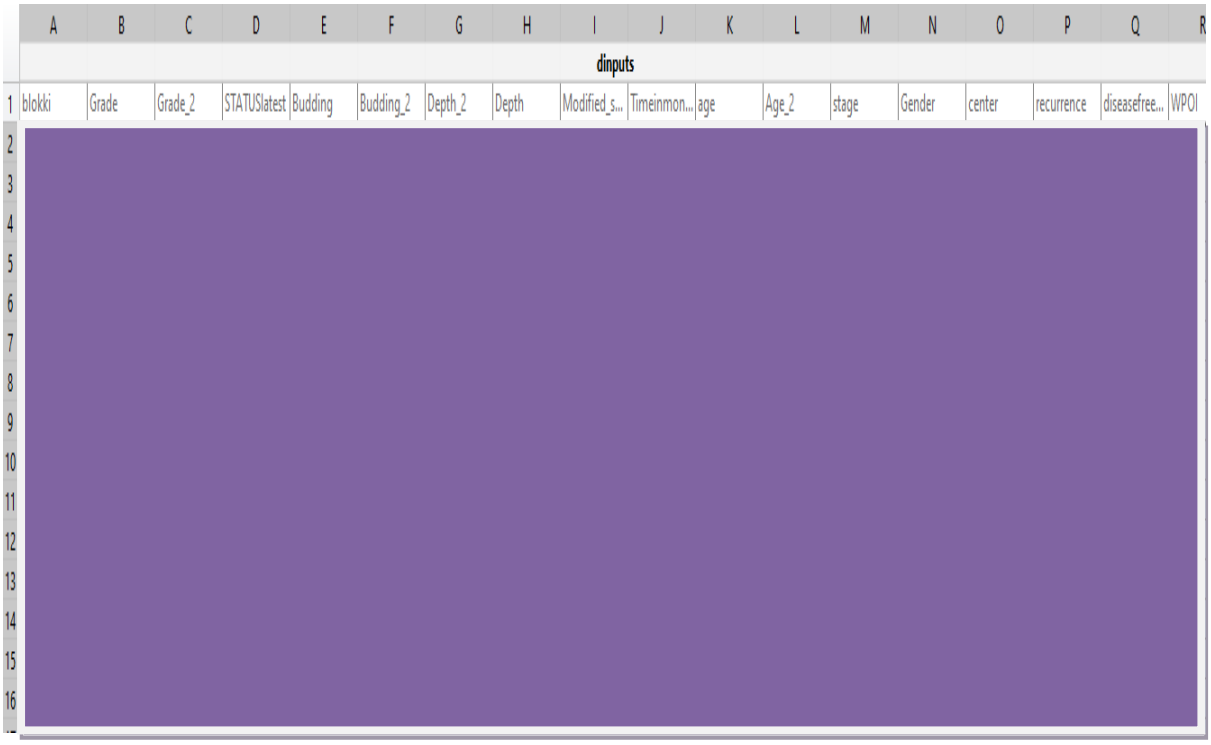
24	0.4280	3	0.8824	-1.1627	-0.4068
25	0.2216	1	0.6502	0.4596	-8.7592
26	0.8231	4	0.1687	1.6523	1.5218
27	0.1100	6	-0.4282	-1.1594	-0.3887
28	0.4323	1	-0.0990	2.9694	-19.4459
29	0.9046	7	0.8234	1.5067	2.0040
30	0.2700	6	0.6896	-0.9032	-1.6793
31	0.9959	1	0.2767	1.1397	1.4440
32	0.1661	1	0.4699	-2.8198	-19.7713
33	0.6651	8	0.9913	3.7482	-14.0066
34	0.8634	8	0.5981	1.0495	7.1251
35	0.0019	3	0.4755	4.8225	-15.6823
36	0.9990	6	0.7029	-1.4873	-6.5347
37	0.8004	5	0.3383	-0.5296	-7.7235
38	0.6454	6	0.8927	-0.7623	-5.7591
39	0.6194	5	-0.3129	4.8481	-8.2967
40	0.4368	8	0.9819	-2.4427	0.1300 C
41	0.1060	6	0.3298	2.3891	0.1300
42	0.1500	2	0.8507	2.1789	-4.5616
43	0.8491	3	0.9570	-2.5124	-9.8823
44	0.8354	4	0.9325	2.4635	3.3405

45	0.1437	4	-0.4915	0.5802	-2.7173
46	0.5130	2	-0.4186	0.3618	-12.2963
47	0.7275	6	0.1901	-0.0832	-8.1574
48	0.6953	6	0.9621	0.4963	-5.4274
49	0.6558	5	0.4024	-3.9396	-0.0728
50	0.8003	1	-0.7277	-1.0807	0.6488
51	0.8370	1	0.9149	4.9301	14.1814
52	0.0622	1	0.9999	-1.2818	-15.8530
53	0.6706	6	0.9871	-0.5121	1.5452
54	0.0927	8	-0.1400	-4.0194	0.9408
55	0.0040	1	0.9980	-4.1313	-17.7367
56	0.2984	6	0.4595	2.0039	-11.7778
57	0.9353	5	0.5278	-3.2787	-0.9055
58	0.7572	2	0.8645	1.9135	2.4817
59	0.5047	1	0.9984	-1.8370	-15.9029
60	0.7270	5	0.2104	-0.6923	-2.8001
61	0.3023	7	-0.3120	-2.3970	-13.4383
62	0.7559	7	0.0304	3.4631	-8.7722
63	0.9406	5	0.5793	1.0317	0.4721
64	0.7649	2	-0.3680	2.3371	-2.9755
65	0.3573	3	0.6706	-2.5627	-2.7173 C

66	0.7981	1	0.8078	-2.6002	-7.0020
67	0.1249	4	0.9430	-1.4899	-1.1581
68	0.2014	1	0.9921	0.6417	0.5212
69	0.5301	1	0.9405	-2.0335	-5.8054
70	0.1993	1	0.3311	-1.7706	-11.4914
71	0.8277	4	0.2922	1.6039	1.2833
72	0.7930	1	-0.1692	0.8879	-7.1476
73	0.3121	8	0.9245	-1.3225	1.6770
74	0.0301	4	0.8749	4.4615	-0.7636
75	0.6352	2	0.3637	-1.8367	-6.6378
76	0.3837	6	0.2904	-3.2736	-9.4573
77	0.9403	4	0.9537	-1.5225	-3.8117
78	0.3088	7	0.9653	4.8740	-8.6108
79	0.6090	7	0.7330	0.1653	0.6459
80	0.6115	2	0.9965	-3.4117	-16.0260
81	0.5786	7	-0.3033	0.0182	-6.1230
82	0.4238	8	0.9708	-2.8854	-4.5458
83	0.3825	6	0.5107	1.1530	-17.6431
84	0.9635	7	-0.3476	0.9206	0.8775
85	0.2728	7	0.2464	-1.7999	-6.0739
86	0.6495	4	0.2909	2.9949	-4.9615

87	0.9125	3	0.0141	-0.0978	0.3055
88	0.8102	1	0.6165	-4.9979	-29.6604
89	0.8845	8	0.9919	-0.9909	4.8790
90	0.1523	4	0.9371	0.3896	-20.7257
91	0.9004	8	0.8799	-1.4484	-0.0272
92	0.4502	5	-0.0496	0.5117	-8.5826
93	0.9015	4	-0.6116	0.8812	-1.2118
94	0.7356	2	0.1969	-1.1840	-0.3601
95	0.2469	2	0.5606	-1.2968	-3.4400
96	0.4739	2	0.6444	4.7693	4.6343
97	0.1539	2	0.9514	4.0626	5.0945
98	0.3370	1	0.2093	2.1260	1.3427
99	0.1097	1	0.4287	1.7429	-14.5608
100	0.8053	7	-0.6022	2.9341	4.8801

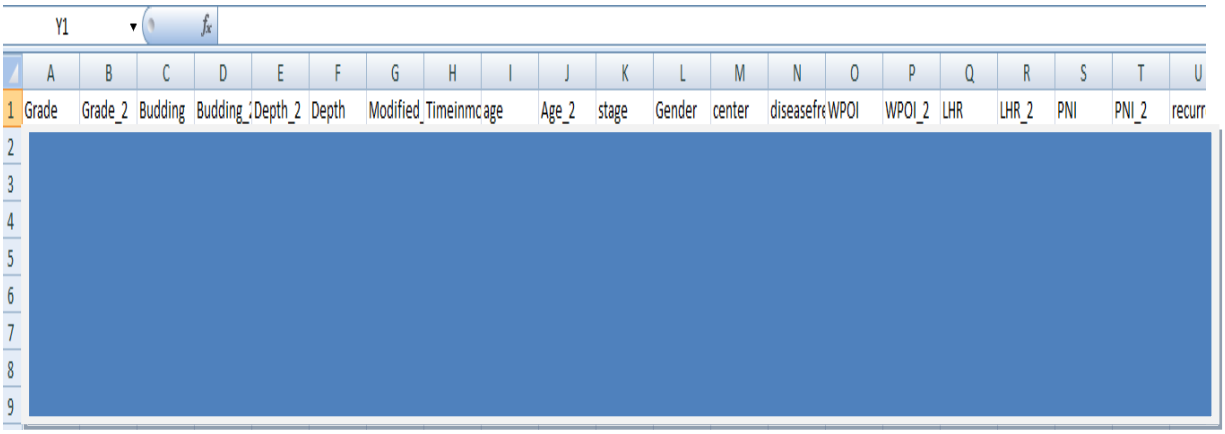
APPENDIX III



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	dinputs																	
1	blokki	Grade	Grade_2	STATUSlatest	Budding	Budding_2	Depth_2	Depth	Modified_s...	Timeinmon...	age	Age_2	stage	Gender	center	recurrence	diseasefree...	WPOI
2																		
3																		
4																		
5																		
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		

.....Details Missing for Ethical and Privacy Concern----------

APPENIDIX IV

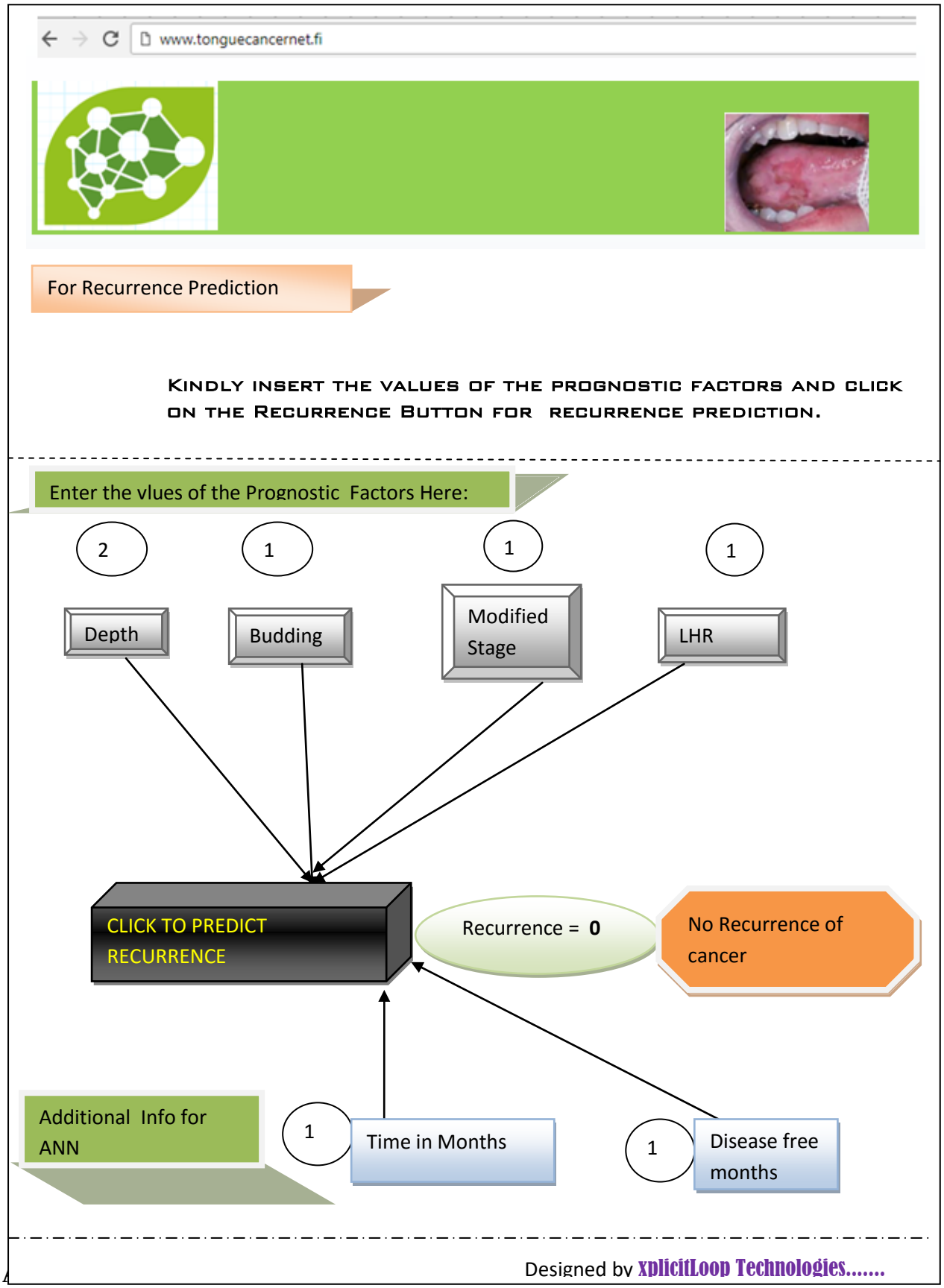


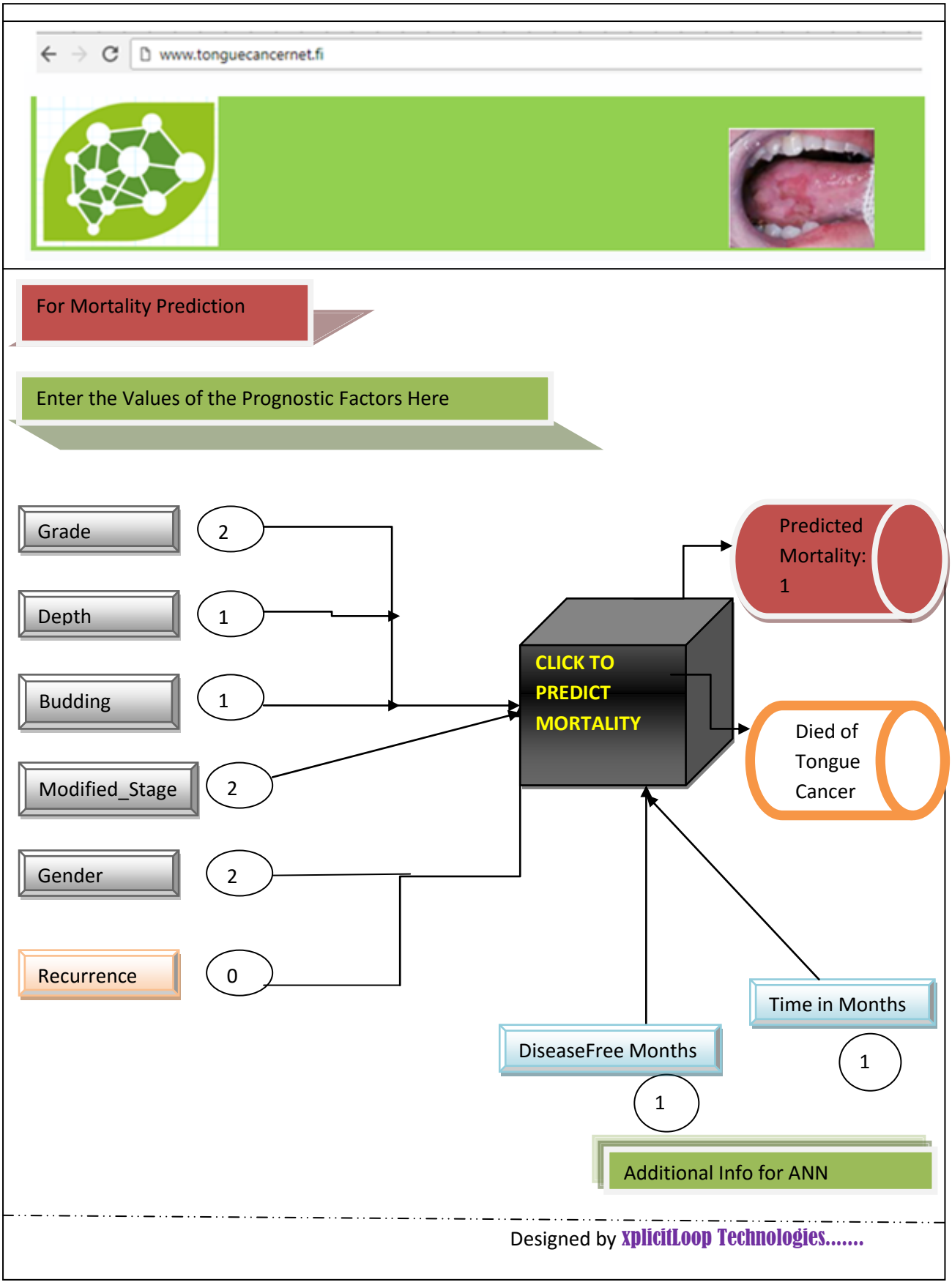
.....Details Missing for Ethical Reasons.----------

APPENDIX V


www.tonguecancernet.fi
.....WELCOME TO TongueCancerNet
TongueCancerNet is an assistive technology based on Artificial Neural Network trained system to predict the Recurrence and Mortality in Tongue Cancer Patients.
KINDLY CLICK ON THE APPROPRIATE BUTTON TO CHOOSE :
Recurrence Prediction:
CLICK ME!
Mortality Prediction:
CLICK ME!

APPENDIX VI


www.tonguecancernet.fi
For Recurrence Prediction
KINDLY INSERT THE VALUES OF THE PROGNOSTIC FACTORS AND CLICK ON THE RECURRENCE BUTTON FOR RECURRENCE PREDICTION.
Enter the vlues of the Prognostic Factors Here:
2
Depth
1
Budding
1
Modified Stage
1
LHR
CLICK TO PREDICT RECURRENCE
Recurrence = 0
No Recurrence of cancer
Additional Info for ANN
1
Time in Months
1
Disease free months
Designed by XplicitLoop Technologies.....


www.tonguecancernet.fi
For Mortality Prediction
Enter the Values of the Prognostic Factors Here
Grade
2
Depth
1
Budding
1
Modified_Stage
2
Gender
2
Recurrence
0
CLICK TO PREDICT MORTALITY
Predicted Mortality: 1
Died of Tongue Cancer
Time in Months
1
DiseaseFree Months
1
Additional Info for ANN
Designed by xplicitLoop Technologies.....