ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY
DEPARTMENT OF CHEMICAL ENGINEERING

MODELING CHEMICAL ENGINEERING PROCESSES USING ARTIFICIAL NEURAL NETWORKS

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTERS OF CHEMICAL ENGINEERING

By
Alemayehu Ambaw

Advisors:

Dr-Ing. Berhanu Assefa
Dr-Ing. Nurelegne Tefera

Addis Ababa University
January 2005

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY
DEPARTMENT OF CHEMICAL ENGINEERING

MODELING CHEMICAL ENGINEERING PROCESSES USING ARTIFICIAL NEURAL NETWORKS

By: Alemayehu Ambaw

Approved By:

1. **Dr.Ing. Berehanu Assefa** Advisor	_____	Signature	_____ Date
2. **Dr.Ing. Nurelegne Tefera** Advisor	_____	Signature	_____ Date
3. **Dr.Ing. Nurelegne Tefera** Chairman	_____	Signature	_____ Date
4. **Dr.Kumudha Raimond** External Examiner	_____	Signature	_____ Date
5. **Dr. Venkata Narasaiah D** Internal Examiner	_____	Signature	_____ Date

Acknowledgment

This is always a very difficult part because so many people have helped in so many different ways. Nonetheless, there are those that should be recognized for the critical help they have provided.

Special thanks to my advisors, Dr-Ing Berhanu Assefa and Dr-Ing Nurelegne Tefera, who have been supporting me from the beginning to the end of this project. Abebe Honelign, who's work turn out to be the pedestal for my thesis work, were there when I needed him most. Also, special thanks to Birihan Haggos who let me use his experimental data on the drying characteristics of potato.

Special thanks to Hiwot Bizuneh, who has been giving me peaceful and tranquil time, effort and opinion all the way through. The encouragement and friendship of contemporaries in the graduate program were critical for me in many respects.

Table of Contents

List of tables

List of Figures

Abstract

In this thesis the application of feed forward type artificial neural networks to model chemical engineering processes are demonstrated with reference to five different problems.

Neural network models are constructed and employed to predict vapor-liquid equilibrium (VLE) data of twelve different binary systems having different chemical structures and solution types (azeotrope-nonazeotrope) in various conditions (isothermal or isobaric). It is observed that the data found by neural network model gives an excellent agreement with the experimental data. In fact the neural network model can be treated as a powerful means for VLE data prediction in a fast and reliable way.

This study has confirmed the feasibility of using a neural network to capture the nonlinear and interacting relationships between the moisture content and different drying conditions of potato. Simulating time series temperature profiles of adiabatic batch reactor has also investigated. Neural network trained with a limited number of experimental data were capable of predicting fresh data that were not used to train the network. The results obtained in using the developed models are physically sound as expected from experience.

Simulating a human operator controlling a chemical plant is also a good instance where the advantage of using artificial neural networks is demonstrated in the thesis. This thesis also describes the use of multilayer feed forward neural networks as a CO_2 analyzer. It was proved that MLP-type network of a relatively simple structure made it possible to predict the CO_2 effluent from a furnace.

Taking in to account the difficulties in experimental conditions, complicated measurements and unavoidable errors of devices used, limit the precision of laboratory measurement results. The accuracy of the results generated by the developed neural network models may be considered satisfactory for engineering calculations.

1 INTRODUCTION

1.1 Overview

From the beginning of digital computing to the end of the 1980s, virtually all data processing applications adopted a basic approach: *programmed computation*. This approach requires the previous development of a mathematical or logical algorithm to solve the problem at hand, which must be subsequently translated into any computational language. This approach is limited, because it can only be used in cases where the processing to be made can be precisely described in a known rule set. Sometimes, however, the development of such a rule set is hard or impossible.

During the late 1980s, a revolutionary approach for data and information processing appeared—*neural networks*. This technique does not require the previous development of algorithms or rule sets to analyze data. This can significantly minimize the software-development work needed for a given application. In most cases, the neural network is previously submitted to a training step using known data; then the methodology necessary to perform the required data processing is extracted. Thus, a neural network is able to identify the required relationships from real data, avoiding the previous development of any model [1].

Neural networks are purely data driven models and have been proven to be universal approximators. In recent years, there has been considerable interest in employing neural networks to model chemical and biochemical processes due to their ability to identify complex input-output relationships [2]. Neural networks are a powerful candidate where it is difficult to use knowledge based models. A model is termed knowledge-based if it has been possible to construct it entirely from prior knowledge and physical insight, or if the physical insight suggests a model structure where the only unknown are parameters which will be estimated from measurements. Knowledge-based models are usually in the state-space form, their state variables and the relationship between them having a physical meaning. If no or insufficient physical insight is available, but only observed

data, no model structure can be defined a priori. In this case, one must resort to black-box models [3].

In the thesis neural networks were used to deal with four different chemical processes modeling problems. Basic steps in neural network design were discussed step by step. Computer programs were written using MATLAB which construct neural network models, compare the performance of competing models so that selection of the best model is possible and use the developed models for the required purposes.

Neural network models were developed: for the prediction of binary VLE data, to analyze the drying characteristics of agricultural product (potato), to simulate a human operator controlling a chemical plant, to analyze CO_2 effluent from a furnace. In the modeling approach of this thesis, the first step was to identify an appropriate neural network topology. Due to its proven functionality and ease of applicability a feed-forward type neural network and specifically *Multiple Layered Perceptrons* (MLPs) were used. The work also addresses the related problems of data pre-processing, feature extraction, architectural design, training and testing techniques.

1.2 The potential of ANNs in modeling chemical engineering processes

The prohibitive expense and enormous time required to develop mathematical models of complex industrial processes from first principles often means that engineers must rely on system identification techniques to develop dynamic process models [4]. Many process monitoring and control schemes are based upon a representation of the relationship between cause and effect variables. In such schemes, this representation is typically approximated using some form of linear models, such as finite impulse response (FIR), auto-regressive with exogenous variable (ARX) and auto-regressive, moving average with exogenous variable (ARMAX) models. Once determined, the process model of the system can be integrated within a variety of process monitoring and control algorithms[5]

Such monitoring and control schemes have found widespread applications in industry and have led to significant improvements in process operations. Unfortunately, the model employed within the schemes tends to be linear in form. Although linear models can provide acceptable performance for many systems, they may be unsuitable in the presence of significant non-linearities. For such system, it may be beneficial to employ a model that reflects the non-linear relationship between cause and effect variables. Preliminary studies have indicated that artificial neural networks (ANNs) may provide a generic, non-linear solution for such systems.

As with standard linear modeling techniques, ANNs are capable of approximating the relationships between cause and effect variables. In contrast to linear techniques however, ANNs offer the benefit of being able to capture non-linear relationships. Since the performance of process monitoring and control algorithms are dependent upon the precision of the model embedded within them, ANN models have the potential to provide benefits to these algorithms when applied to non-linear systems.

During the past years, neural networks have been widely used in chemical engineering. First attempt appeared about decade and a half ago [6]. The application of neural networks in various areas of chemical process engineering, such as fault detection,

diagnosis, process control, process design, and process simulation has been presented. In ISI (The Institute for Scientific Information, at location: http://www.isinet.com/) database, which contain published articles, more than 1000 articles with the keyword "chemical process modeling using neural networks" are found. These numbers are even more overwhelming considering that different communities apply different names to the subject.

In the field of phase equilibria, the use of neural networks as a part of or complete predictive tool for vapor-liquid equilibrium (VLE) prediction, representing different separate phase equilibrium methods is mentioned. Piotrowski, et al. [6] used MLP type neural network to model VLE data in the urea synthesis process. They used *DynaMind* program to create the network and to train and test the network.

Another example is the use of neural network method as an alternative tool to help an engineer to choose a suitable phase equilibrium method for further process calculation. For instance, a paper by Oreski, et al.[7], can be mentioned where neural networks were trained with inputs describing several combinations of physical properties associated with the corresponding methods of phase equilibrium. This model helps an engineer to choose a suitable phase equilibrium method for further process calculation. The Kohonon type neural network was used to classify the collected samples into none, one or more possible method of phase equilibrium and estimate the reliability of the proposed classes (adequacy of different methods of phase equilibrium). From the numerous phase equilibrium methods, in their paper they choose fifteen of the most often used in practice. Kohonen map with almost clearly separated clusters of vapor, vapor/liquid and liquid phase regions and 15 probability maps for each of the specific phase equilibrium method, were obtained. In there analysis of the results, they confirmed the hypothesis that the use of Kohenen neural networks for separation of the classes was correct.

On the other hand, this thesis attempts to show the potential advantages of using a trained neural network as a new method of VLE data prediction method. The proposition is that, a purely data driven and black-box modeling technique like artificial neural networks

provide a better handiness of VLE related problems. The thesis aimed to show the ease and simplicity of using artificial neural networks in the field of phase equilibria in general.

In the field of drying, the work by Kamenski and Tomczack [8] presented applications of artificial neural networks in evaluation of drying kinetics of selected materials. RBF, which are important for the description of a process dynamics, are presented. As an example, they considered drying and degradation of ascorbic acid in agricultural products are considered. Calculations concerning the prediction of moisture content and temperature of dried material in time for tested products were made using RBF. Inputs to the network were drying process parameters: air flow rate V_{G0} and drying agent temperature at the inlet T_{G0} to the apparatus, material temperature $T_m(i)$ and material moisture content $X_m(i)$ in the moment before the predicted output from the network. The output from the network included material temperature and moisture content in time "i + 1" as related to the input values. Very good agreement of experimental data and values calculated was reported.

Another paper by Rodriguez, et al. [9] presented a dynamic model for a drum dryer using neural networks and a system of linear first order differential equations. The model obtained was a combination of neural network and a system of differential equations. The efficiency of this hybrid model and a linear differential model were compared with experimental data obtained for milk drying. The hybrid model represented adequately the evolution of final temperature of the dry product with a smaller error than that of the differential model.

In the approach of this thesis MLP type neural networks were developed to analyze the drying characteristics of locally obtained potato. Calculations concerning the prediction of moisture content of the material in time were made. Inputs to the network were drying process parameters: air flow rate V_G and drying agent temperature at the inlet T_G to the apparatus, air relative humidity RH, initial moisture content X_{m0} as well as the total

drying time i where we want to do the prediction. The output from the network is the moisture content of the material as related to the input values.

The thesis also includes a work on simulating a human operator controlling a chemical plant. The same problem was addressed by Kevin [10] using Fuzzy logics programmed in MATLAB. The fuzzy logic based controller simulates the human operator excellently. The aim of this thesis is to address the same problem by the use of artificial neural networks.

Al-Duwaish, et al.[11] showed how to use artificial neural networks as a process analyzer. In their paper, artificial neural networks, which are known for their ability to model nonlinear systems and their inherent noise-filtering abilities, are used as O_2 analyzer to predict O_2 concentration in a boiler. The main idea of the neural-based O_2 analyzer is to infer the O_2 content in the flue gas from other easily measured process variables. In the area of process analyzer this thesis aimed to show how to construct and use neural network models for the purpose of predicting the CO_2 effluent from a furnace. The neural network model developed in this thesis was designed to predict the next value in a series from a fixed number of previous values (looking ahead a single time step). When the next value in a series is generated, further values can be estimated by feeding the newly-estimated value back into the network together with other previous values: time series projection.

Detailed steps in designing neural network models and related issues were plainly incorporated in the thesis. Commercially available neural network software packages can be used to apply this new computational method for the modeling requirement. The price to purchase such package may be too expensive for a country like Ethiopia. So, the MATLAB files developed in this thesis may be helpful for those who planed to apply this technique.

1.3 Tasks and methodology of the thesis

"Possibilities abound in geologic, climatic, meteorologic, personality, cultural, historical, spectral, electromagnetic, and other data, as well as from microscopic images of cells to macroscopic images of regions of the earth obtained from satellite scans and radio telescope images of galaxies. There remains only for the researcher in some area to glean the essentials and begin to explore the classification and recognition of patterns in data that will lead to discoveries of associations and cause-effect relationships. The association between patterns and their causes are the bricks from which the wall of scientific knowledge is built" Carl G. Looney [12]

To this end, the general goal of this thesis work can be stated simply as:

> *To be equipped with and to demonstrate the highly praised information processing paradigm called artificial neural networks and to show the advantages of applying this method in the activity of picking out the association between patterns and their causes in selected areas of chemical process engineering.*

By taking some modeling problems the thesis aimed to demonstrate the different steps in the activity of developing a neural network models and to confirm advantages of using this new way of computational tool. The following list shortly discusses the problems sited in the thesis and gives a general outline of the methodology followed in addressing the problems using artificial neural networks. A detailed description of the methodology is given for each problem in the corresponding chapter where each of them is addressed separately.

- **Neural network models for prediction of VLE data:** This section aim to describe a predictive modeling approach based on neural networks which offer a useful alternative to the conventional thermodynamic based prediction methods. Binary VLE data are collected from different data sources. After collecting the data the appropriate data pre-processing procedures are applied to prepare the data for the purpose of network training. Two different approaches were considered

regarding this problem. The first approach is to develop a neural network based VLE prediction model for a given binary mixture using collected VLE data at different constant pressures and temperatures.

The second approach is to group together VLE data of different binary mixtures and see if it is possible to get a single generalized neural network model to predict the VLE data for the group in general.

- **Neural network model of a drying process:** This section aimed to demonstrate how to model a drying characteristic of potato based on drying rate data obtained from a laboratory experiment.

- **Neural network model to predict the time series temperature profile of adiabatic batch reactor:** Here a feed forward type artificial neural network is sought to predict the time series temperature profile of adiabatic batch reactor. Experimental time series temperature profile data obtained for three different initial temperatures of the reacting materials were used to model the neural network. The developed model is supposed to predict the time series temperature profiles for initial temperature conditions other than that obtained in the laboratory experiment.

- **Neural network model to simulate a human operator controlling a chemical plant:** This aimed to design a neural network model which simulates a human operator controlling a chemical plant. The network construction is based on data about how a human operator use input parameters to decide a set point to a conventional PID controller in a chemical plant.

- **Neural network model as a CO_2 analyzer:** The objective is to predict ahead the value of CO_2 that varies in time using previous values of CO_2 in the effluent and values of gas flow rates to a furnace.

1.4 Outline of the thesis

Chapter 2 takes a closer look at artificial neural networks. Artificial neural networks are defined and the different network structures are introduced. It also discusses the important topic of activation functions.

Chapter 3 provides steps in designing neural networks. Data collection and pre-processing are other important topics in chapter 3. This chapter covers the supervised training of feed-forward neural networks via a particular type of training algorithm known as the Levenberg Marquardt algorithm.

Chapter 4 is concerned with application of artificial neural networks to selected chemical engineering processes. Detailed descriptions on building neural network models are given. Computer programs written in MATLAB are used to deal with the selected problems and the outputs from such models are shown and discussed in this chapter.

Chapter 5 is the appendix chapter. Summary of the neural network models developed for the twelve binary mixtures are given in this chapter. This part also presents the programming codes of MATLAB script and function files used to create, train, test and use neural network models in the thesis.

2 BASICS TO ARTIFICIAL NEURAL NETWORKS

This chapter provides an introduction to artificial neural networks. The novice in artificial neural networks may want to read this chapter first. The expert may skip reading it as it comprises no material necessary for the continuity of the rest of the report. An artificial neural network as a system is defined in section 2.1. Section 2.2 introduces the fundamental unit of a neural network- a neuron. Finally section 2.3 discusses how the network gets more complex by arranging neurons in a network of layers.

2.1 Artificial neural networks defined

In its most general form, a neural network is a machine that is designed to *model* the way in which the brain performs a particular task or function of interest; the network is usually implemented using electronic components or simulated in software on a digital computer. To achieve good performance, neural networks employ a massive interconnection of simple computing cells referred to as "neurons" or "processing units." The following definition is given by Aleksander and Morton [13].

A neural network is a massively parallel distributed processor that has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:

1) Knowledge is acquired by the network through a learning process.
2) Interneuron connection strengths known as synaptic weights are used to store the knowledge.

The procedure used to perform the learning process is called a *learning algorithm,* the function of which is to modify the *synaptic weights* of the network in an orderly fashion so as to attain a desired design objective. Neural networks are also referred to in the literature as neural computers, connectionist networks, parallel distributed processors, etc. [13]

2.2 Model of a neuron

Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, a typical neuron collects signals from others through a host of fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin stand known as an *axon*, which splits into thousands of branches. At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

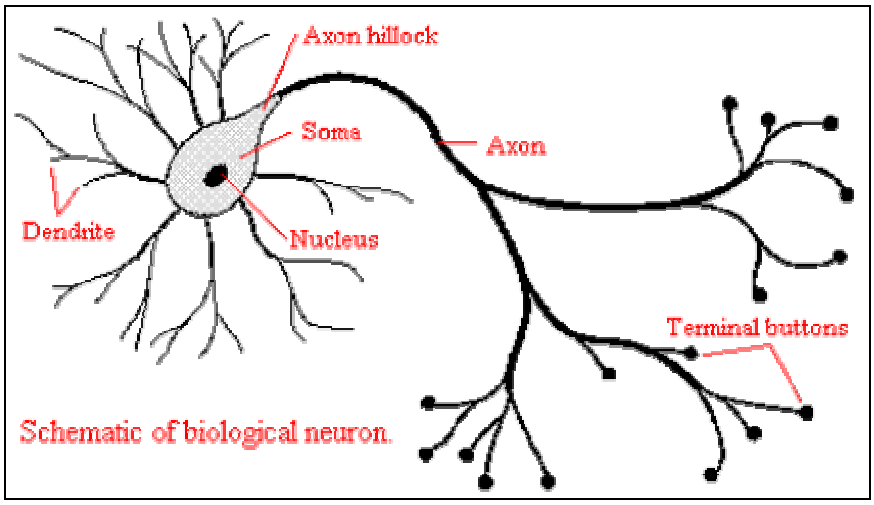

Figure 2-1 Components of biological neuron

In simulating biological neurons, first the essential features of neurons and their interconnections are deduced. Then typically program a computer to simulate these features. However because our knowledge of neurons is incomplete and our computing power is limited, our models are necessarily gross idealizations of real networks of neurons.

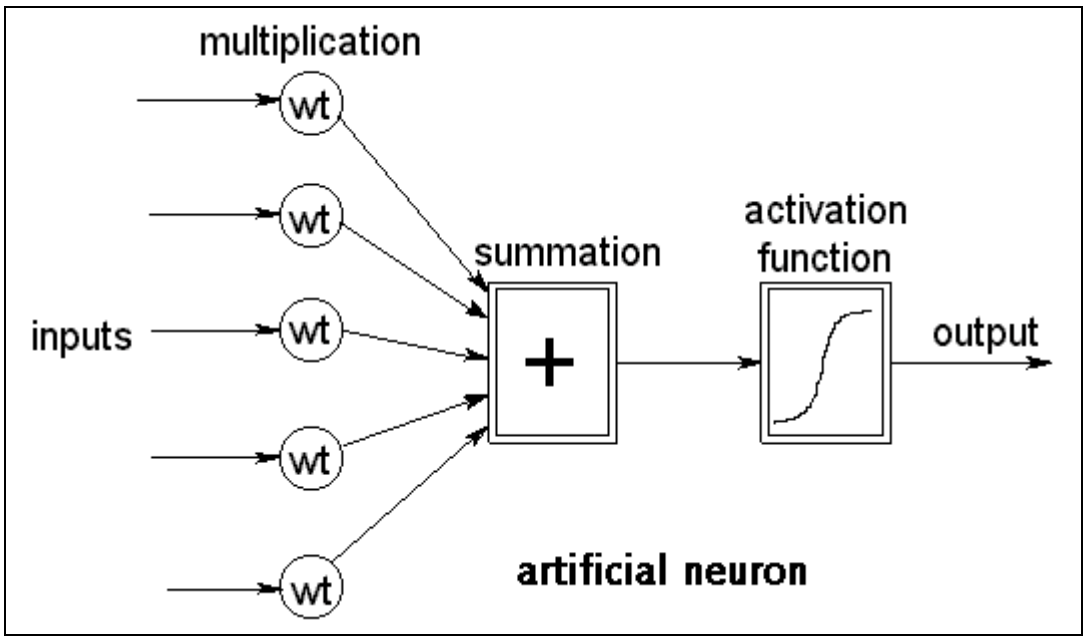

Figure 2-2 Model of artificial neuron

A *neuron* is an information-processing unit that is fundamental to the operation of a neural network. Figure 2.2 shows the *model* for a neuron. We may identify three basic elements of the neuron, as described here [13]:

1. A set of *synapses* or *connecting links,* each of which is characterized by a weight or *strengths* of its own.
2. An *adder* for summing the input signals, weighted by the respective synapses of the neuron; the operation described here a *linear combiner.*
3. An *activation function* for limiting the amplitude of the output of a neuron. the activation function is also referred to in the literature as a *squashing function* in that it squashes (limits) the permissible amplitude range of the output signal to some finite value.

A detailed mathematical model of a neuron is shown in Figure 2.3. Models may include an externally applied *threshold* that has the effect of lowering the net input of the activation function. On the other hand, the net input of the activation function may be

increased by employing a *bias* term rather than a threshold. In mathematical terms, we may describe a neuron ***m*** by writing the following pair of equations:

$$r_m = \sum_{n=1}^{N} (w_{mn} x_n + b_m) \tag{2.1}$$

and $$y_m = f(r_m) \tag{2.2}$$

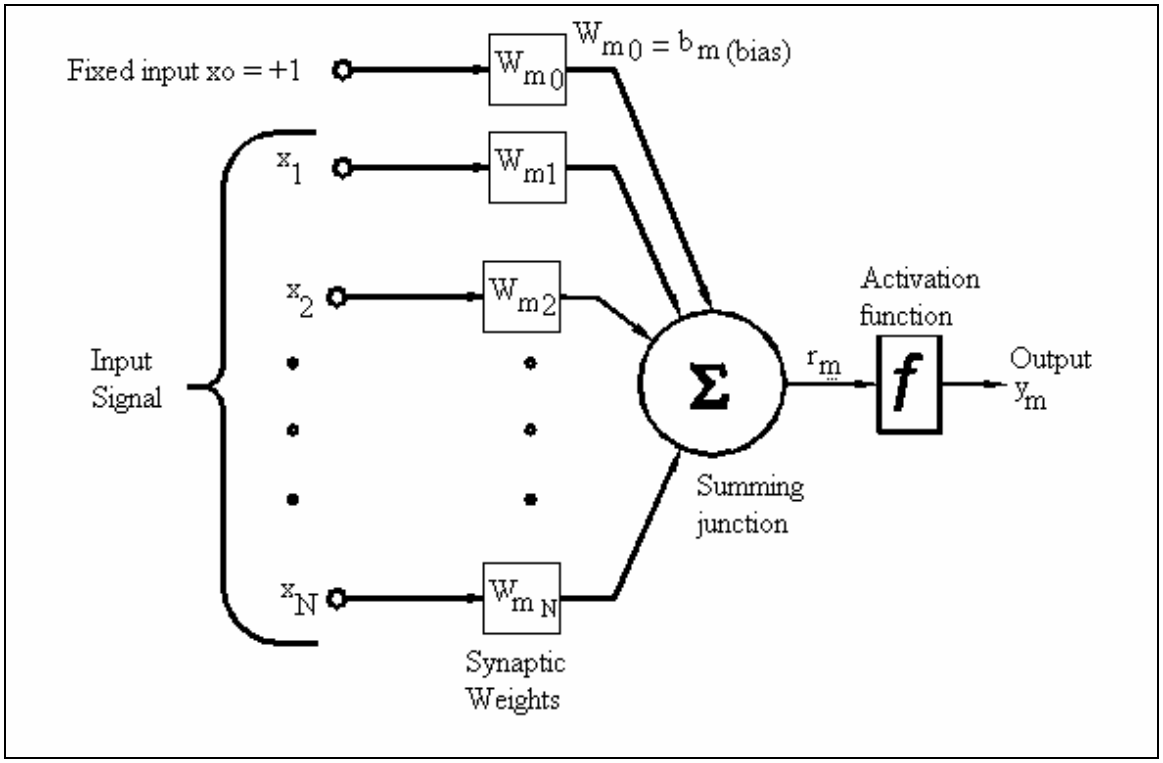

Figure 2-3 Mathematical model of a neuron

Where $x_1, x_2, \ldots, x_N$ are the inputs; $w_{m1}, w_{m2}, \ldots, w_{mN}$ are the synaptic weights of neuron m; r_m is the linear combiner output; b_m is the bias term; f is the activation function; and y_m is the output signal of the neuron.

The scalar input x is transmitted through a connection that multiplies its strength by the scalar weight **w**, to form the product wx, again a scalar. The neuron shown in Figure 2.3 has a bias b_m but bias term may not be used depending on the situation. You may view the bias as simply being added to the product wx as shown by the summing junction or as

shifting the function f to the left by an amount b_m. The bias is much like a weight, except that it has a constant input of 1. The transfer function net input r_m, again a scalar, is the sum of the weighted input wx and the bias b_m. This sum is the argument of the transfer function f which takes the argument r_m and produces the output y_m. Note that w and b are both adjustable scalar parameters of the neuron.

2.3 Feed-forward networks of layered neurons

The manner in which the neurons of a neural network are structured is intimately linked with the learning algorithm (rule) used to train the network. Learning algorithms are discussed in chapter 3. The following sections briefly discuss the operation of MLP type neural networks.

The power of single neuron can be greatly amplified by using multiple neurons in a network of layered connectionist architecture. Neurons layered in such a way is also called *feed-forward artificial neural network* and abbreviated to FANN. Feed-forward artificial neural networks include MLPs, Functional Link Networks (FLNs) and Radial Basis Function Networks (RBFNs)[12].

MLPs are perhaps the most popular network architecture in use for many problems. This is the type of network shown in Figure 2-4. The network has a simple interpretation as a form of input-output model, with the weights and thresholds (biases) the free parameters of the model. Such networks can model functions of almost arbitrary complexity, with the number of layers, and the number of units in each layer, determining the function complexity. Important issues in MLP design include specification of the number of hidden layers and the number of units in these layers [14]

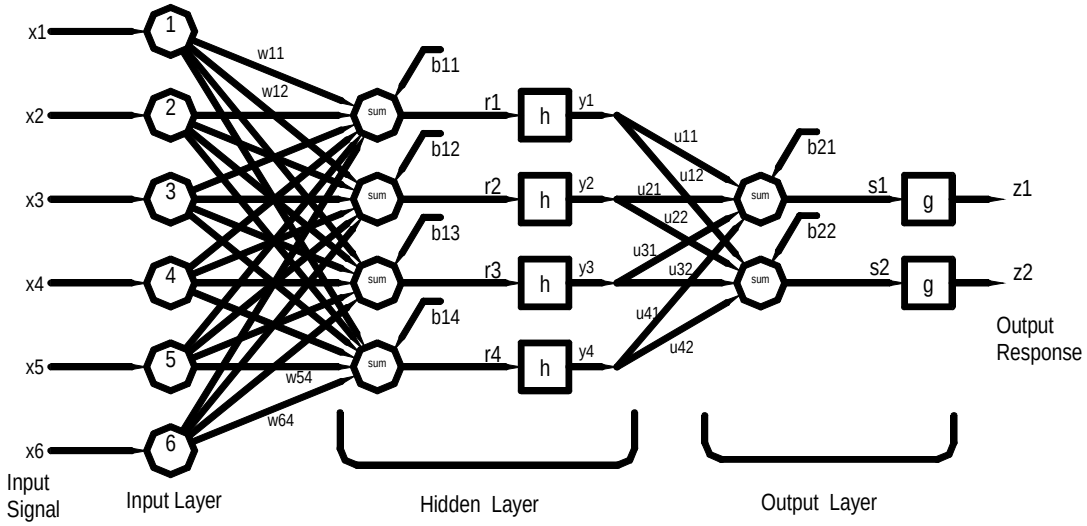

Figure 2-4 A representation of a simple 3-layer feed-forward ANN

On the left is the layer of inputs, or *branching,* nodes, which are not artificial neurons. A feature vector $\mathbf{x} = (x_1,\ldots, x_N)$ that represents a pattern enter the input layer on the left with each component x_n entering one and one input node. From each $\boldsymbol{n}^{\text{th}}$ input (branching) node, the nth component x_n fans out to each of the $\boldsymbol{M}$ neurons in the middle layer. Thus each $\boldsymbol{m}^{\text{th}}$ hidden (middle) neurons has a fan-in of all $\boldsymbol{N}$ input components. As each $\boldsymbol{x_n}$ enters the $\boldsymbol{m}^{\text{th}}$ neuron of the hidden layer, it is modified via multiplication by the *synaptic weight* $\boldsymbol{w_{nm}}$ for that connection line. All resulting products $\boldsymbol{w_{nm}x_n}$ at the $\boldsymbol{m}^{\text{th}}$ hidden neuron are summed over n to yield

$$r_m = \sum_n w_{nm} x_n \tag{2.3}$$

and $y_m = h(r_m)$ (2.4)

is the activation output. Doing the same for the output layer we have

$$s_j = \sum_m u_{mj} y_m \tag{2.5}$$

and $z_j = g(s_j)$ (2.6)

is the network output.

2.4 Activation functions

Activation functions for the hidden units are needed to introduce nonlinearity into the network. Without nonlinearity, hidden units would not make nets more powerful than just plain perceptrons (which do not have any hidden units, just input and output units). The reason is that a linear function of linear functions is again a linear function. However, it is the nonlinearity (i.e, the capability to represent nonlinear functions) that makes multilayer networks so powerful [15]. Almost any nonlinear function does the job, except for polynomials. For back propagation learning, the activation function must be differentiable, and it helps if the function is bounded; the sigmoidal functions such as logistic and tanh and the Gaussian function are the most common choices [15].

Threshold transfer function

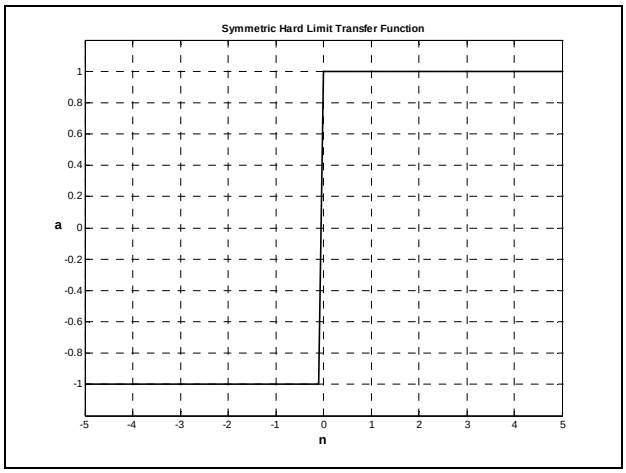

Figure 2-3 Threshold transfer function

The threshold transfer function gives a perceptron the ability to classify input vectors by dividing the input space into two regions. Specifically, outputs will be 0 if the net input n is less than 0, or 1 if the net input n is 0 or greater.

Log-sigmoid transfer function

For hidden units, sigmoid activation functions are usually preferable to threshold activation functions [15]. Networks with threshold units are difficult to train because the

error function is stepwise constant, hence the gradient either does not exist or is zero, making it impossible to use backprop or more efficient gradient-based training methods. With sigmoid units, a small change in the weights will usually produce a change in the outputs, which makes it possible to tell whether that change in the weights is good or bad. With threshold units, a small change in the weights will often produce no change in the outputs.

$$a = \log sig(n) = \frac{1}{(1+\exp(-n))} \tag{2-7}$$

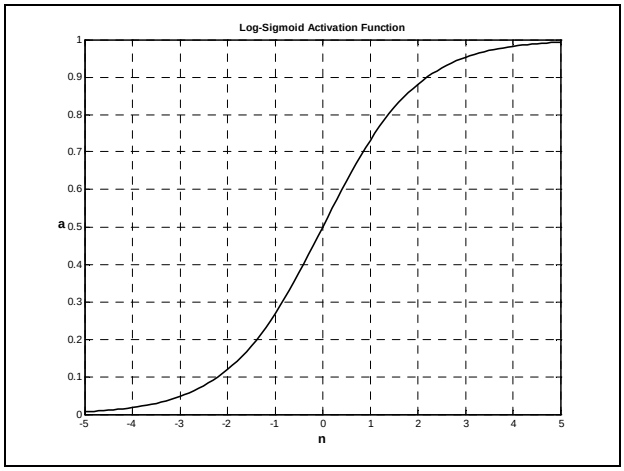

Figure 2-5 Log sigmoid activation function

Log-sigmoid generates outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity.

Tan-sigmoid transfer function

Alternatively, multilayer networks may use the tan-sigmoid transfer function.

$$a = \tan sig(n) = \frac{2}{(1+\exp(-2n))} - 1 \tag{2.8}$$

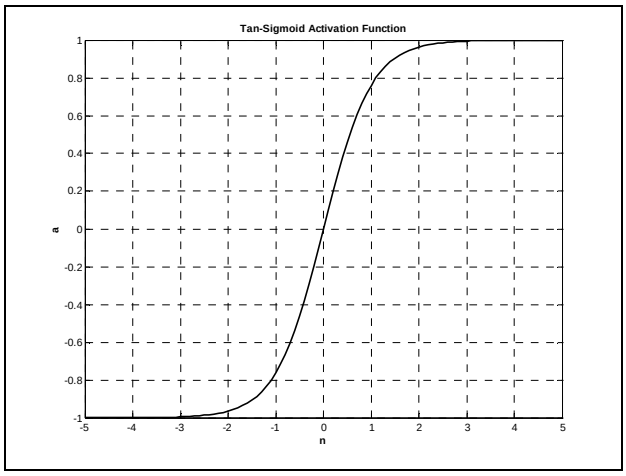

Figure 2-6 Tan-sigmoid activation function

Linear transfer function

Occasionally, the linear transfer function is used in backpropagation networks.

$$a = purelin(n) = n \tag{2.9}$$

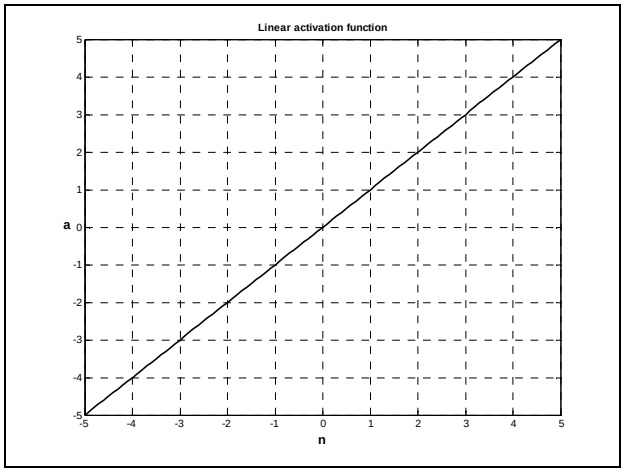

Figure 2-7 Linear activation function

3 DESIGNING ARTIFICIAL NEURAL NETWORKS

The goal of this chapter is to present some basic steps in designing a neural network for a given problem. There are generally four steps in network designing process:

1. Assemble and pre-process raw data to get training data
2. Create the network object
3. Train the network
4. Simulate the network response to new inputs

Neural network design requires a lot of hard work, collected raw data are pre-processed before using them in network training, network object is created, performances are monitored, parameters adjusted, connections added, rules modified, and on and on until the network achieves the desired results.

Section 3.1 describes basic issues around data pre-processing. Then section 3.2 gives some answers for questions around network construction. Simulation and training of the network objects are presented in section 3.3.

3.1 Data pre-processing

The object of data pre-processing is to produce the training set of the NN, which represents the relationship of network inputs and outputs. Preprocessing means that the existing data is processed (in some way) before the network is trained on it. By preprocessing the data, the problem may be much more suitable for the network. The major tasks in data preprocessing are: [16]

- Data cleaning
- Data integration
- Data transformation
- Feature extraction

3.1.1 Data cleaning

Many data sets are imperfect due to the presence of missing values and noise in the data. Incomplete data comes from

- data value not available when collected
- different consideration between the time when the data was collected and when it is analyzed.
- human/hardware/software problems

Noisy data comes from the process of data

- collection
- entry
- transmission

Inconsistent data comes from

- Different data sources
- Functional dependency violation

To handle data imperfection, data cleaning algorithms must be developed which can.

- Fill in missing values
- Identify outliers and smooth out noisy data
- Correct inconsistent data
- Resolve redundancy caused by data integration

The various methods that can be used to deal with data cleaning requirements are thoroughly discussed in issues around data mining. In this section a program is written to identify outliers and remove them before the data is used to train a network.

3.1.2 Data integration and data transformation

Data integration is a process of combining data from multiple sources into a coherent store. Data integration may need several issues to be considered. For the same real world entity, attribute values from different sources are different due to different representations, different scales etc. Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve data quality.

The primary purpose of data transformation is to modify the distribution of input variables so that they can better match outputs. The performance of a neural network is often improved through data transformations. For MLPs, we map each $\boldsymbol{x_n^{(q)}}$ in to the range [β , $1-\beta$], because MLPs do not train properly on 0s and 1s as inputs [12]. The linear transformation is usually used as shown in Equation (3.1).

$$L_n(x)=(1-2\beta)\frac{(x-x_{\min(n)})}{(x_{\max(n)}-x_{\min(n)})}+\beta \tag{3.1}$$

So that $L_n(x_{\min(n)})=\beta$ and $L_n(x_{\max(n)})=1-\beta$. As Looney [12] suggested a good value of β is 0.2 (or 0.15).

The nature of data distribution in the collection set may dictate the appropriate transformation function. For instance, in case there are small values disproportionately close together and the larger values are spread out (for fixed n), the logarithmic transformation function G_n is suitable [12], where

$$G_n(x)=(1-2\beta)\frac{\ln\left(\frac{(x-x_{\min(n)})}{(x_{\max(n)}-x_{\min(n)})}+1\right)}{\ln(2)}+\beta \tag{3.2}$$

For large value close together and small values farther apart, the exponential transformation function X_n is suitable [12].

$$X_n(x)=(1-2\beta)\frac{\left((1-\exp[-\frac{(x-x_{\min(n)})}{(x_{\max(n)}-x_{\min(n)})}])\right)}{(1-\exp[-1])}+\beta \tag{3.3}$$

After performing the appropriate pre-processing steps on the raw data, the next step is feature extraction or feature selection. In feature extraction important attributes that have great influence in describing the input/output relationships are selected and used to train the network. This step removes redundant and irrelevant information that could cause

extraneous noise and degrade the performance of the network. One of the best feature extraction methods is discussed in the next section.

3.1.3 Feature extraction

Usually a large set of attributes are collected about a given input/output mapping problem. This is to get a good set of attributes powerful enough to glean out the required pattern between the inputs and the outputs. There are several techniques to extract feature out of a pre- collected data set. Correlations of features, weighing features by importance, the Belue-Bauer method of feature extraction are some of the widely used techniques.

The method discussed by Belue and Bauer [12] used the MLP itself in the selection of those features that makes a significant contribution to the capability of the neural network to approximate the function required in prediction. In this method, first the MLP train on the full set of *N* features on the total exemplars. Then the training is used to determine the relative significance of the input features. Using this result the method suggests eliminating the ones that have low significance. The tool used a *relevancy metric*. The simple relevancy metric is defined on the *n*th input component x_n by

$$\eta_n = \sum_{m=1}^{M} w_{nm}{}^2 \tag{3.4}$$

where w_{nm} is the weight on the line from the n^{th} input node to the m^{th} hidden neuron of an MLP with a single hidden neuron. The idea here is that the sensitivity of the network output to each input feature is determined and the input feature are ranked in order $x_{n(1)}$, …, $x_{n(N)}$. The features on the low end of the ordering may be eliminated, provided that their relevancies are sufficiently low compared to those of the important features [12].

3.2 Creating network structure

After getting the training data set, the neural network can be built. To do this the network structure should be defined first. Defining the network structure includes:

- Network structure selection
- Sizing the network structure
- Training the neural network

3.2.1 Structure selection

There is no well defined procedure or rule to be used in building a neural network. Data preprocessing – structure selection – network sizing – network training steps are interrelated and the designer need to establish methods to develop a set of competing neural network models and a way of performance measure to select the best one.

Artificial neural networks are a set of several different models featuring a wide variety of different architectures, learning strategies and applications. At present several types of networks specialized in carrying out various tasks are distinguished. Table 3.1 categorizes the different types of artificial neural networks.

Category 1 networks are the most powerful, versatile, and reliable nonlinear classifier-recognizers [12].The networks in Category 2 may be trained to some extent to adjust the field of attraction for the different classes, but are not yet sufficiently reliable or efficient. Category 3 networks are self-organizing and perform linearly separable clustering of data. The nature of the problem we are trying to solve determines which neural network will be employed.

Table 3-1 Hierarchy of Artificial Neural Networks

CATEGORY	TYPE	NAME OF NETWORK TYPE
1	Feedforward(FANNs)	MLPs (multiple-layered perceptrons) FLNs (functional link networks) RBFNs (radial basis function networks) LVQNs (learning vector quantization networks)
2	Recurrent(RNNs)	Amari networks (parallel) Hopfield networks (random serial)
3	Self-organizing maps(SOMs)	Kohonen's SOFMs (self-organizing feature maps) Sprecht's probabilistic networks Bezdek's fuzzy c-means networks Hybrid learning vector quantization networks Grossberg's ART networks (adaptive resonance theory) SOLVQNs (self-organizing LVQNs)

MLPs are perhaps the most popular network architecture in use today; this network has a simple interpretation as a form of input-output model, with the weights and biases the free parameters of the model. Such networks can model functions of almost arbitrary complexity. Largely most researchers who use artificial neural networks usually use the feed forward type and MLPs in particular [12]. Basing network topology selection on this confirmed fact, this thesis applies MLP type neural networks to different chemical engineering modeling problems. Network sizing and network training, discussed in the following section are mainly concerned with issues related to MLP.

3.2.2 Sizing the network structure

Let ***N*** be the number of input branching nodes, ***M*** be the number of hidden neurons, ***J*** the number of output neurons, ***Q*** be the number of exemplar vectors for training. The network architecture is determined by the numbers ***N, M,*** and ***J***. The main questions in sizing a neural network are:

- How many layers of neurons to use?
- How many input nodes to use?
- How many neurons in the hidden layers should we use?
- How many neurons should we use in the output layer?

The number of layers to use is provided by the Hornik-Stinchcombe-White result stated as [12].

> *A feed-forward artificial neural network with two layers of neurodes and non-constant non decreasing activation function at each hidden neurode can approximate any piecewise continuous function from a closed bounded subset of Euclidean N-dimensional space to Euclidean J-dimensional space with any prespecified accuracy, provided that sufficiently many neurodes be used in the single hidden layer.*

The data determines ***N***, ***J*** and ***Q***. The number ***N*** of input nodes must be the number ***N*** of feature in the feature vectors, so that once a set of feature is chosen, the number ***N*** is fixed. ***J*** will fix the number of output neurons.

The remaining task in network sizing is to set the number of hidden neurons- **M**. Unfortunately, there is currently no universal guideline for determining the optimal number of hidden neurons. The selection of the number of hidden neurons is often the result of empirical experimentation combined with trial and error [2].

3.3 Training the network

Once a network has been structured for a particular application, that network is ready to be trained. There are two approaches to training - supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help.

The vast bulk of networks utilize supervised training. Unsupervised training is used to perform some initial characterization on inputs. However, in the full blown sense of being truly self learning, it is still just a shining promise that is not fully understood, does not completely work, and thus is relegated to the lab [17].

3.3.1 Supervised training

In this approach, both the inputs and the required target outputs are provided to the network. Then the network which is now under the training mode processes the inputs and generates predicted values as output from the network. By comparing the resulting outputs against the desired outputs it calculates the errors. Errors are then propagated back through the system, causing the system to adjust the weights and biases which control the network. This process occurs iteratively as the adjustable parameters of the network are continually adjusted to minimize the prediction error. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined.

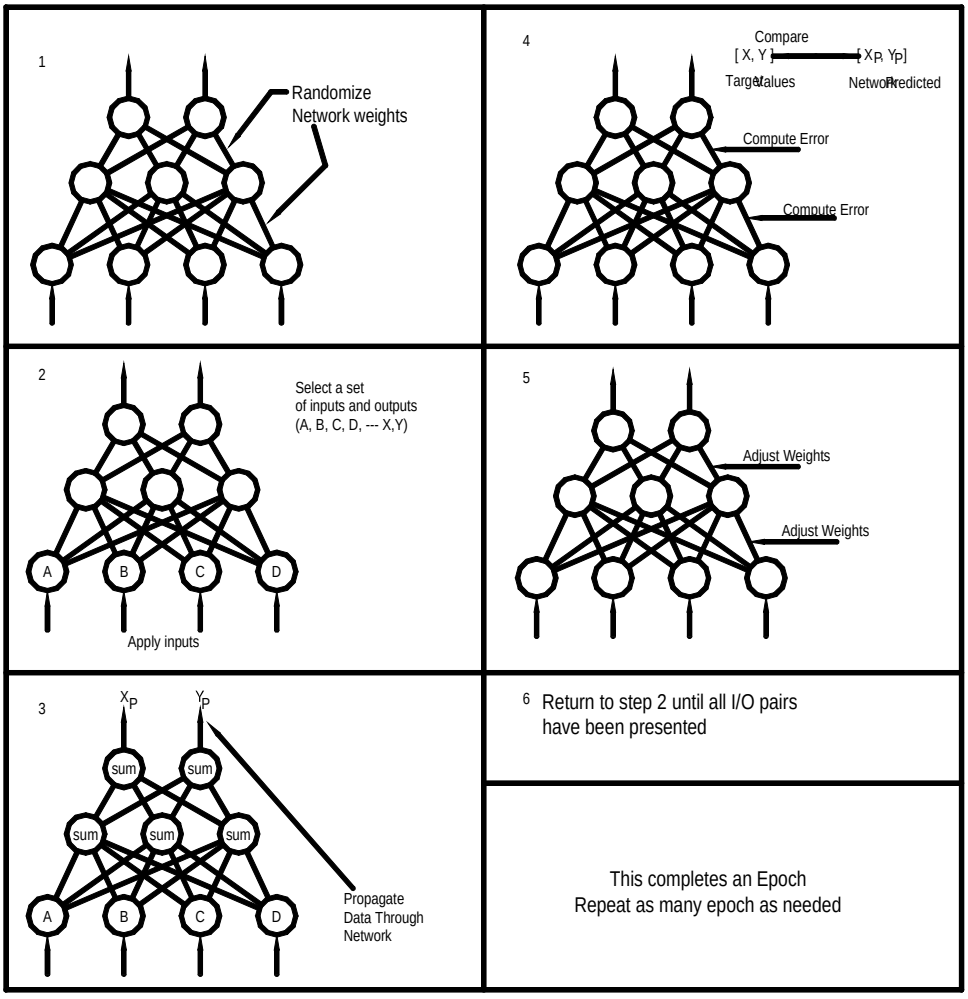

Figure 3-1 Supervised training using the backpropagation algorithm

Ideally, there should be enough data so that part of the data can be held back as a test. Usually before starting the training, we need to select disjoint subsets of exemplar pairs of size $\boldsymbol{Q_T}$ for training, $\boldsymbol{Q_{val}}$ for validation of the training, and $\boldsymbol{Q_{ver}}$ for verification of the model, $\boldsymbol{Q_T + Q_{val} + Q_{ver} = Q}$ where $\boldsymbol{Q}$ is the total number of exemplars.

When using the different training algorithms, a decision must be made on when to stop the training process. Some guides are really important to perform systematic network training and the list given by Bishop [18] is a good start.

1. Stop after a fixed number of iterations. The problem with this approach is that it is difficult to know in advance how much iteration would be appropriate, although an approximate idea can be obtained from some preliminary tests. If several networks are being trained (e.g. with various numbers of hidden neurons) then the appropriate number of iterations may be different for different networks
2. Stop when a predetermined amount of CPU (central processing unit) time has been used. Again, it is difficult to know what constitutes a suitable time unless some preliminary tests are performed first. Some adjustment for different architectures may again be necessary.
3. Stop when the error function falls below some specified value. This suffers from the problem that the specified value may never be reached; some limit on the CPU time may also be required.
4. Stop when the relative change in error function falls below some specified value. This may lead to premature termination if the error function decreases relatively slowly during some part of the training run.
5. Stop training when the error measured using an independent validation starts to increase. This approach is generally used as part of a strategy to optimize the generalization performance of the network.

In practice some combination of the above methods may be employed as part of a largely empirical process of parameter optimization.

Sometimes a network simply can't solve the problem at all, in this case the designer then has to review the different parts of the network, the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training functions, and even the initial weights themselves. Those changes required to create a successful network constitute a process wherein the "art" of neural networking occurs [13].

3.3.2 Backpropagation algorithm: The Levenberg-Marquardt method

The backpropagation supervised learning algorithm is used to find weights in multilayer feedforward networks. The backpropagation algorithm is conceptually simple. Following each input data Vector, the network performance is evaluated on the target values in the validation set. The errors resulting from the comparison of the actual and target output values are propagated backward through the network, and the weight values are adjusted to minimize error. With respect to neural networks, the performance criterion is the minimization of squared error. Therefore, the total system error is expressed as follows:

$$E = \frac{1}{2}\sum (\varepsilon^n)^2 = \frac{1}{2}\|\varepsilon\|^2$$

Where, ε^n is the error of the *n*th pattern, and ε is a vector with elements ε^n.

The problem of learning in neural networks is formulated in terms of the minimization of the error function ***E***. This error is a function of the adaptive parameters (weights and biases) in the network. Many learning laws are in common use. Most of these laws are some sort of variation of the best known and oldest learning law, Hebb's Rule. Research into different learning functions continues as new ideas routinely show up in trade publications. Some researchers have the modeling of biological learning as their main objective. Others are experimenting with adaptations of their perceptions of how nature handles learning. Either way, man's understanding of how neural processing actually works is very limited. Learning is certainly more complex than the simplifications represented by the learning laws currently developed [17]. The backpropagation algorithm is the most practical and commonly used model for neural networks. Some of the well known types of backpropagation algorithms are [19]:

- *Gradient descent backpropagation*: is a network training function that updates weight and bias values according to gradient descent
- *Gradient descent with adaptive learning rate backpropagation* : is a network training function that updates weight and bias values according to gradient descent with adaptive learning rate
- *Gradient descent with momentum & adaptive learning rate backpropagation*: is a network training function that updates weight and bias values according to gradient descent momentum and an adaptive learning rate.
- *Levenberg-Marquardt backpropagation*: is a network training function that updates weight and bias values according to Levenberg-Marquardt optimization.

In this section a brief sketch of the Levenberg-Marquardt (**LM**) training algorithm is given. The emphasis to this training algorithm is due to its confirmed advantages over the other methods. The discussion is adopted from Bishop [18]. Consider the sum-of-squares error function in the form

$$E = \frac{1}{2}\sum (\varepsilon^n)^2 = \frac{1}{2}\|\varepsilon\|^2 \tag{3.5}$$

Where ε^n is the error of the n^{th} pattern, and ε is a vector with elements ε^n Suppose we are currently at a point $\mathbf{W}_{\text{old}}$ in weight space and we move to a point $\mathbf{W}_{\text{new}}$. If the displacement $\mathbf{W}_{\text{new}}$ - $\mathbf{W}_{\text{old}}$ is small then we can expand the error vector ε to the first order in the Taylor series

$$\boldsymbol{\varepsilon}(\boldsymbol{W}_{new}) = \boldsymbol{\varepsilon}(\boldsymbol{W}_{old}) + \boldsymbol{J}(\boldsymbol{W}_{new} - \boldsymbol{W}_{old}) \tag{3.6}$$

Where we have defined the matrix **J** with elements

$$(\boldsymbol{J})_{ni} \equiv \frac{\partial \boldsymbol{\varepsilon}^n}{\partial \boldsymbol{w}_i} \tag{3.7}$$

The error function Equation (3.5), can be rewritten as

$$\boldsymbol{E} = \frac{\mathbf{1}}{\mathbf{2}}\left\|\boldsymbol{\varepsilon}(\boldsymbol{W}_{old}) + \boldsymbol{J}(\boldsymbol{W}_{new} - \boldsymbol{W}_{old})\right\|^2 \tag{3.8}$$

If we minimize the error with respect to the new weight $\mathbf{W}_{\text{new}}$ we obtain

$$\boldsymbol{W}_{new} = \boldsymbol{W}_{old} - (\boldsymbol{J}^T \boldsymbol{J})^{-1} \boldsymbol{J}^T \boldsymbol{\varepsilon}(\boldsymbol{W}_{old}) \tag{3.9}$$

For the sum-of-error function, Equation (3.6), the elements of the Hessian matrix take the form

$$(H)_{ik} = \frac{\partial^2 E}{\partial w_i \partial w_k} = \sum_n \left\{ \frac{\partial \varepsilon^n}{\partial w_i} \frac{\partial \varepsilon^n}{\partial w_k} + \varepsilon^n \frac{\partial^2 \varepsilon^n}{\partial w_i \partial w_k} \right\} \tag{3.10}$$

If we neglect the second term, the Hessian can be written in the form

$$\boldsymbol{H} = \boldsymbol{J}^T \boldsymbol{J} \tag{3.11}$$

In principle the update formula, Equation (3.9), could be applied iteratively to try to minimize the error function. The problem with such an approach is that the step size which is given by Equation (3.9) could turn out to be relatively large, in which case the linear approximation, Equation (3.6),on which it is based would no longer be valid. In the Levenberg-Marquardt algorithm this problem is addressed by seeking to minimize the error function while at the same time trying to keep the step size small so as to ensure that the linear approximation remains valid. This is achieved by considering a modified error function of the form

$$\tilde{\boldsymbol{E}} = \frac{1}{2} \left\| \boldsymbol{\varepsilon}(\boldsymbol{W}_{old}) + \boldsymbol{J}(\boldsymbol{W}_{new} - \boldsymbol{W}_{old}) \right\|^2 + \lambda \left\| \boldsymbol{W}_{new} - \boldsymbol{W}_{old} \right\|^2 \tag{3.12}$$

where the parameter λ governs the step size. For large value of λ the value of $\left\| W_{new} - W_{old} \right\|^2$ will tend to be small. If we minimize the modified error Equation (3.12) with respect to W_{new}, we obtain

$$\boldsymbol{W}_{new} = \boldsymbol{W}_{old} - (\boldsymbol{J}^T \boldsymbol{J} + \lambda \mathbf{I})^{-1} \boldsymbol{J}^T \boldsymbol{\varepsilon}(\boldsymbol{W}_{old}) \tag{3.13}$$

where **I** is the unit matrix.

In practice a value must be chosen for λ and this value should vary approximately during the minimization process.

4 APPLICATIONS

4.1 Introduction

In this thesis, MLP type neural networks are used to model five different chemical engineering problems:

1. VLE data prediction model for binary mixtures
2. Modeling the drying characteristic of potato
3. Modeling the temperature-time profile of an adiabatic reaction
4. Simulating a human operator controlling a chemical plant
5. Modeling a CO_2 effluent analyzer

Though each problem has its own characteristic features, developing a neural network model in each case include the following steps:

- **Step 1:** Data for the stated problem is collected from relevant data source. Data integration requirements are done manually before using the MATLAB programs. The collected and integrated data is then stored in a separate data file.
- **Step 2:** Data transformation is done before starting the network training. The pre-processed data is divided in to three different sets, training set (50%), testing set (25%) and validation set (25%).
- **Step 3:** The developed MATLAB program COMPARE.m, do data transformation, network construction, network training, and selecting the best model.
- **Step 4:** MATLAB program EXTRACT.m, Further analyze the performance of the model selected in step 3, then the neural network description is extracted and saved in a separate file.
- **Step 5:** MATLAB program CONSTRUCT.m, use the network parameters extracted in Step 4 to put the selected model in its operating mode.
- **Step 6:** USE.m, is written to use the developed neural network model to do the required analysis work.

The computer program COMPARE.m which is stated in Step 3, construct a set of competing neural network models. After training them simultaneously it compare the performance of each model. Graphical outputs from this computer program are used to select the best neural network topology.

The neural network model comparison is mainly used to choose the optimum number of neurons in the hidden layer and the type of activation functions to use in each layer. This program let the user to do an experiment on the effect of changing different network parameters like:

- Number of neurons in the hidden layer.
- Data transformation.
- Transfer function in the hidden layer.
- Transfer function in the output layer.
- Training algorithm.
- Parameters of the given training algorithm.

Network parameters and related issues in using computer programs developed in this thesis work are discussed in detail and is given in Appendix 6.2.1.

The performance of a neural network during training is measured based on the mean squared error as shown in section 3.3.2. After completing a training step the overall performance of a model is measured. Using only numeric values of error functions are usually deceptive in judging the goodness of a model. So, in the neural network modeling approach of this thesis, graphical outputs of error distributions were also used to help the model selection step. Average absolute errors (AAE), Percent absolute errors (PAE) and Regression R- values (RRV) were the error functions used in judging the performance of a model. The mathematical form of this error functions is given below for clarity.

The average absolute error is obtained by summing the absolute value of all the errors and dividing the total sum by the number of data points.

$$AAE = \frac{(\sum_{q=1}^{Q} abs(t_q - n_q))}{Q} \tag{4.1}$$

where, t_q is the target value for data number q, n_q is the neural network predicted value for data number q, Q is the total number of data and AAE is the average absolute error.

The Regression R-value is a value obtained from correlation analysis between model predictions and target values.

The percent absolute error is calculated during each prediction with the following equation

$$PAE = (\frac{abs(t_q - n_q)}{t_q}) \times 100 \tag{4.2}$$

comprehensive description of the problems and outputs obtained by using the developed computer programs are discussed in the following sections.

4.2 Neural network base VLE prediction models

Vapor-liquid equilibrium (VLE) calculations are one of the fundamental calculations typically done by a chemical engineer, applicable in a wide variety of unit operations (e.g. flash drums, and distillation columns design) and critical for properly designing the size, specifications, and operating temperature/pressure of all unit operations.

The simplest model for VLE relating the liquid and vapor compositions in equilibrium is Raoult's Law:

$$y_i P = x_i P_i^* \tag{4.3}$$

where y_i is the mole fraction of component "i" in the gas phase, P is the total pressure, x_i is the mole fraction of component "i" in the liquid phase, and P_i^* is the vapour pressure of component "i" at the specified temperature.

The vapor pressure of component "i" can be calculated through the empirical Antoine's Equation:

$$P_i^* = 10^{\left(A_i - \frac{B_i}{T - C_i}\right)} \tag{4.4}$$

where A_i, B_i, and C_i are empirical constants specific to the identity of component "i" (tabulated in Felder and Rousseau and the Parameters worksheet in this spreadsheet), T is the temperature in degrees Celsius, and P_i^* is the vapour pressure of component "i" in mm Hg or torr

Although Raoult's Law is extremely simple, it is extremely powerful for at least estimating the VLE behavior of, in particular, two similar liquids at moderate temperatures and pressures. Most typically, Raoult's Law is used to calculate the bubble-point temperature/pressure or dew-point temperature/pressure of a liquid mixture.

Even for the simplest model like Rault's, an iterative algorithm is required to solve the different VLE problems. For instance, if we want to calculate the bubble-point temperature, given the liquid composition and the total pressure first we need to estimate the bubble temperature, next using this estimate we calculate the first-pass values of P_i* using Equation 4.4, then we calculate the total pressure resulting from this initial estimate. Finally revising the temperature estimate and re-iterating until the calculated total pressure matches the actual total pressure. Such iterative steps are common for the different thermodynamic based VLE prediction models.

Unfortunately, when two components with significant intermolecular interactions are used or when higher pressures and/or temperatures are involved, Raoult's Law breaks down and we need more sophisticated models which take into account the non-ideality of the vapor and liquid phases. In general, VLE in these cases can be calculated using the modified Raoult's Law equation:

$$y_i \Phi_i P = x_i \gamma_i P_i^* \tag{4.5}$$

where Φ_i accounts for the gas-phase non-ideality and γ_i accounts for the liquid-phase non-ideality.

Φ_i can be calculated via equations of state relationships for non-ideal gases.Gases tend to behave relatively ideally at the lower pressures and moderate temperatures typical for VLE problems with organic chemicals in industry, making this non-ideality less important to consider. However, liquid phase non-ideality γ_i is important at all temperatures and pressures depending on the identity of the two chemicals used and can make a significant difference to the calculated phase behavior. Therefore, we need a way to calculate these γ_i values.

In practice, experimental equilibrium data are correlated by some of the well known Van Laar, Margules, Scatchard Hamer, Redlich-Kister, Wilson, Whol, Hala and other equations. These means that constant and the interaction terms of the equations, which best fit the selected algebraic equations on the data points, are determined. Then, the

vapor liquid equilibrium data are calculated with the most appropriate constants in such equations, are used in design calculations for rectification and distillation columns.

The above brief discussion may depict the complexity and difficulty in using the conventional VLE calculation models. The way we develop neural network models and the ease of using the developed models may create a good place for researchers working around VLE problems.

A number of industrially important processes bring two phases into contact. When the phases are not in equilibrium, mass transfer occurs between the phases. The rate of transfer of each species depends on the departure of the system from equilibrium. Quantitative treatment of mass-transfer rate requires knowledge of the equilibrium states (T, P, and compositions).

Thus the neural network based VLE data prediction models developed in this thesis were designed in consideration of these forms of VLE problems. There are two main tasks covered in relation to VLE prediction models:

1. The first task is to develop a neural network based VLE prediction model for a given binary mixture using the collected VLE data at different equilibrium conditions
2. The second task is to group together VLE data of different binary mixtures and see if it is possible to get a single generalized neural network model to predict the VLE data for the group in general.

Section 4.2.1 will discuss results obtained in the activity of neural network modeling for twelve different binary systems. By taking one of the binary VLE data the section demonstrate how the VLE prediction is modeled using artificial neural networks. Section 4.2.2 will show the result obtained in developing a generalized neural network model that can be used for a group of binary mixtures.

4.2.1 ANN based VLE data prediction for selected binary mixtures

VLE data at different constant pressures and temperatures are collected for 12 different binary mixtures. These data are obtained from ECDB--*The Engineering Chemistry Data Base*, developed by the Laboratory of Computer Chemistry (LCC), the Institute of Chemical Metallurgy (ICM), and Chinese Academy of Sciences (CAS) [21]. After collecting the VLE data the following two steps were followed to prepare them for the network training.

Although VLE problems with other combinations of variables are possible, those of engineering interest are usually dew point or bubble point calculations; there are four classes [20]

- BUBL P: Calculate $\{y_i\}$ and P, given $\{x_i\}$ and T
- DEW P: Calculate $\{x_i\}$ and P, given $\{y_i\}$ and T
- BUBLE T: Calculate $\{y_i\}$ and T, given $\{x_i\}$ and P
- DEW T: Calculate $\{x_i\}$ and T, given $\{y_i\}$ and P

1. Data integration: here the different data at different conditions are integrated in to a single set giving equilibrium pressures (P), liquid compositions (x), vapor compositions (y) and the equilibrium temperatures (T)

2. Based on the type of VLE problem to be addressed the collected data are arranged in to input/ output format as shown in Figure 4-1. Each binary system needs four different models. So, the total number of neural network model developed for this case are 48.

Computer programs used to construct and train neural network models are given in the appendix.

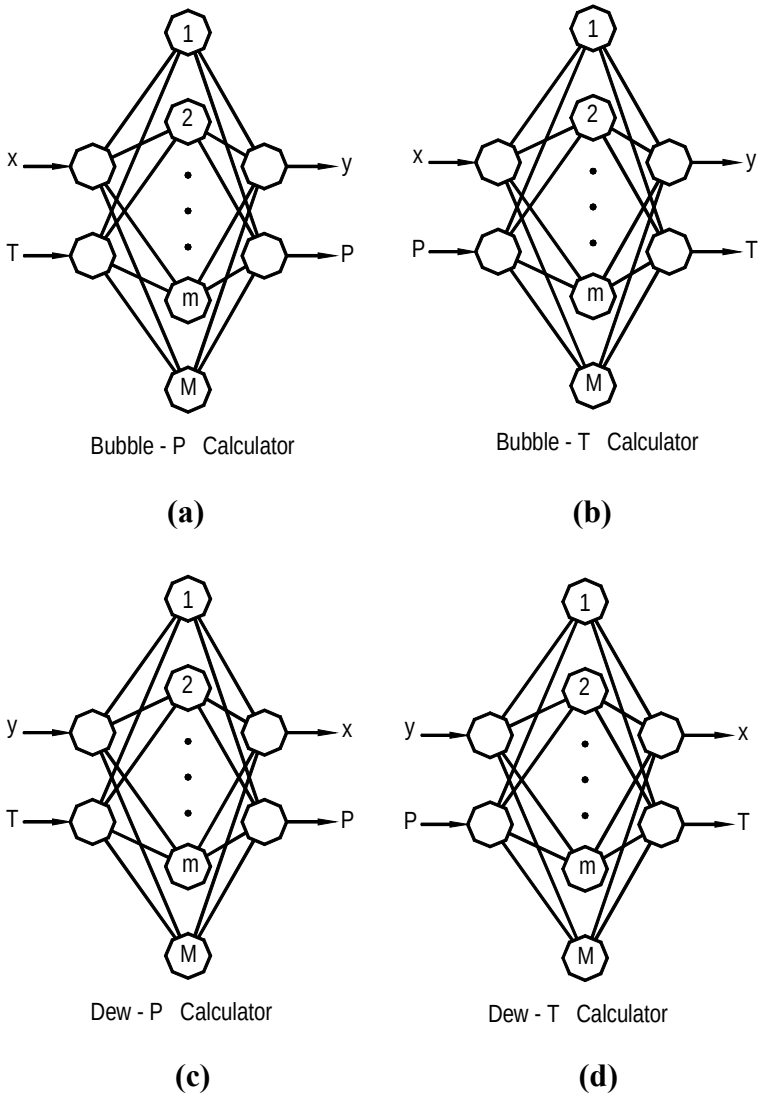

Figure 4-1 Input/Output neural network models for the four classes of VLE problems.

Results obtained by following the above steps are demonstrated in the following sections by taking *Ethanol-Benzene* as a case example. For Ethanol – Benzene the data collection and integration steps results in a single data set with the following data ranges.

- Temperature 293.15 K to 462.05 K
- Pressure 8.4 kPa to 1480.3 kPa

4.2.1.1 Bubble-P calculation

In this section a neural network model that determines bubble composition and bubble-point pressure, given liquid composition and bubble-point temperature, is designed. Graphical representation of the model is shown in Figure 4-1 (a).

The first step in the computer program COMPARE.m is to load the data into the MATLAB workspace and arrange them into the proper input/output format as shown in Figure 4-1 (a). The data transformation is linear using Equation (2.6) shown in Chapter 2. To insure the ability of the network model to predict unseen data, the initially collected VLE data set was split into three groups. So, the next step in the computer program is to divide the data up into training, validation and test subsets. The program takes one fourth of the data for the validation set, one fourth for the test set and one half for the training set.

We are now ready to create a network and train it. In sizing the neural network to use for the problem, the input vectors (equilibrium temperature (T) and liquid composition(x)) set the number of input nodes required to two. The output vectors (the bubble composition (y) and the bubble pressure (P)) set the number of neurons in the output layers to two.

The remaining task in network sizing is to set the number of hidden neurons. Unfortunately, there is currently no universal guideline for determining the optimal number of hidden neurons. The selection of the number of hidden neurons is often the result of empirical combined with trial and error. With a 2-**M**-2 topology, where **M** is the number of neurons in the hidden layer, the number of neurons **M** was varied and the performance of the topology was measured. So the program creates a set of neural network models with different number of neurons in the hidden layer. The performance of the different models was assessed using different performance measures. The average absolute errors and the regression R-values of the different neural network models in competition were used to select the best one.

Training algorithms, activation function in the hidden layer, activation function in the output layer, bias in the hidden layer, bias in the output layer, number of training epochs, parameters of the selected training method and sometimes the data transformation steps are the different constraints we set in the program before starting the network training. Then the program trains the neural networks in succession under the setting and calculates the performance of each network topology and gives the result in graphical form as shown in Figures 4-2 and Figure 4-3.

The Levenberg-Marqurdt backpropagation algorithm was the fastest and the one which gives best performance. Using sigmoid type activation function in the hidden neurons and linear type in the output neurons give best results. Using sigmoid type activation function in both layer were also comparable. Biases are essential in both layers and the performances of neural networks without bias were not good enough. Results from running the computer program COMPARE.m are shown in the following figures.

In selecting the best model, the performance results obtained in predicting both the bubble pressures and the bubble compositions must be taken together. Performance measures based on the average absolute error and the regression R-value in predicting both the bubble pressure and the bubble compositions should be analyzed to select the best neural network topology.

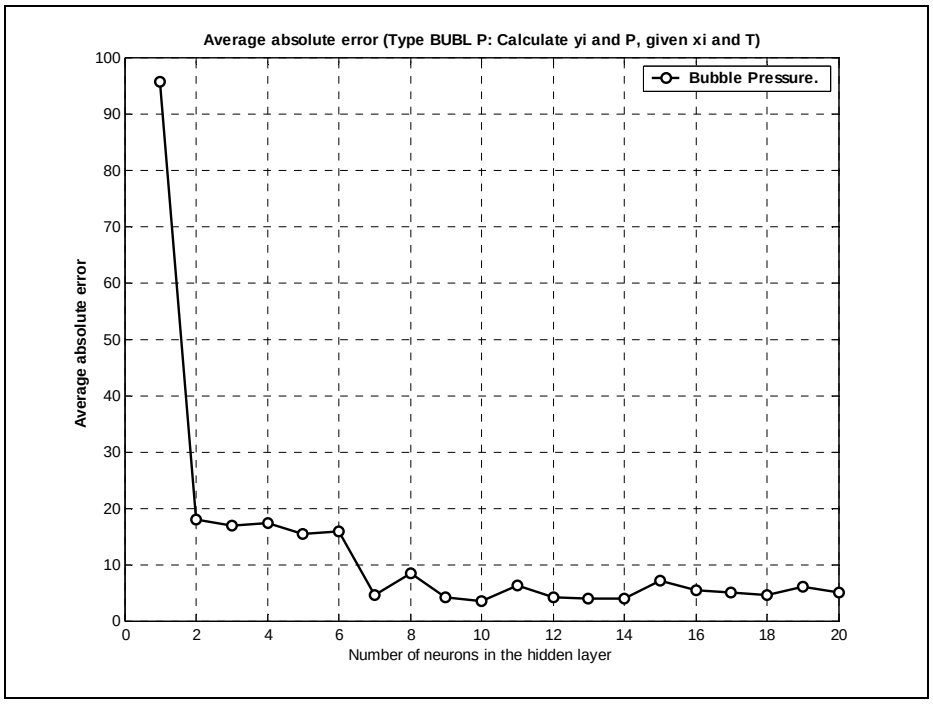

Figure 4-2 Average absolute error based performance measures, *Bubble-P calculation, Predicting the bubble-point pressure, Ethanol _Benzene*

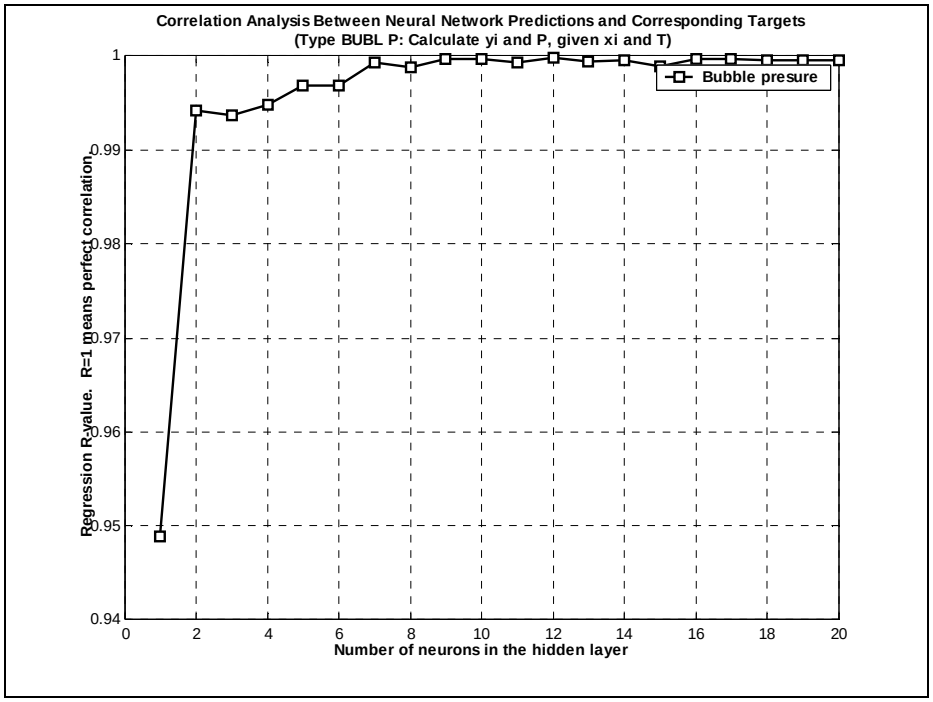

Figure 4-3 Regression R-value based performance measures, *Bubble-P calculation, predicting the bubble-point pressure, Ethanol _Benzene*

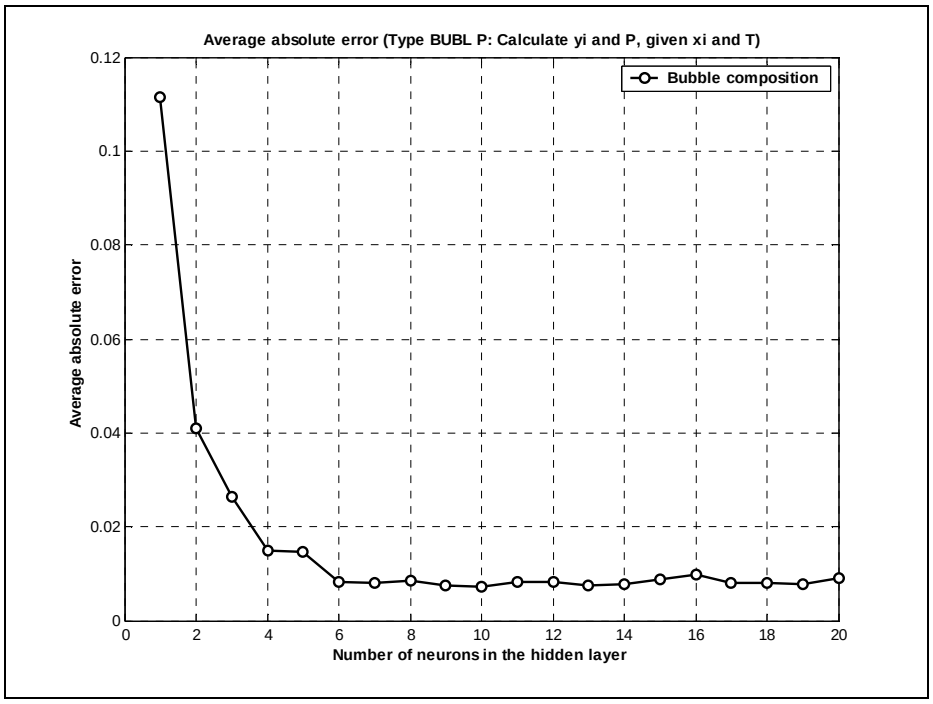

Figure 4-4 Average absolute error based performance measures, *Bubble-P calculation, predicting the bubble composition, Ethanol _Benzene*

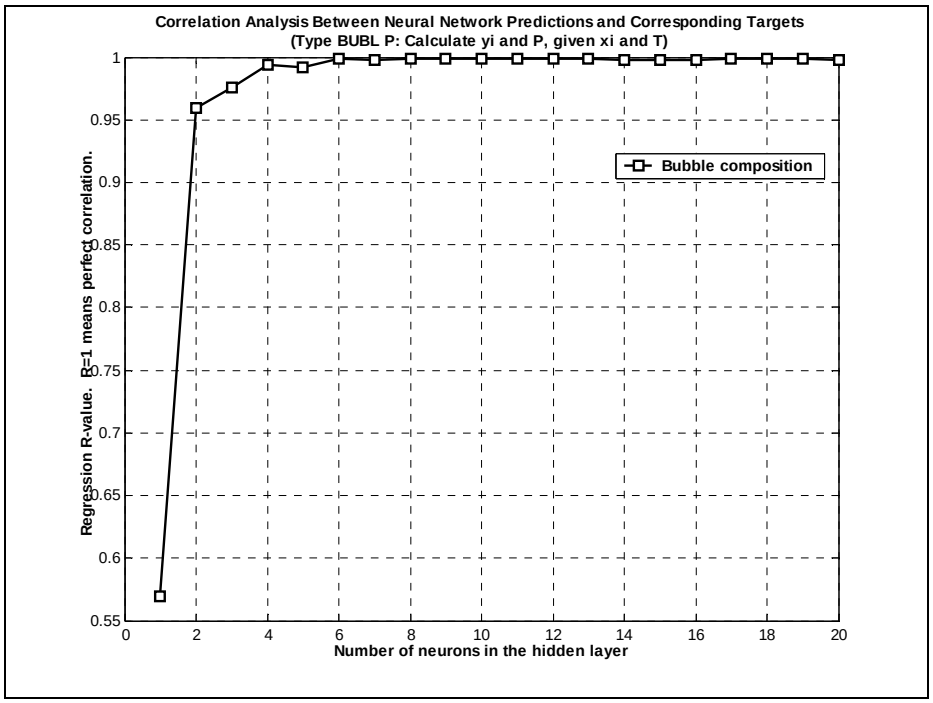

Figure 4-5 Regression R-value based performance measures, *Bubble-P calculation, Predicting the bubble composition, Ethanol _Benzene*

Selecting the best neural network topology was made by a careful analysis of the above four figures. Starting from Figure 4-5, which gives the regression R-values in predicting the bubble compositions, we can see that using hidden neurons exceeding six give no further improvement. This fact is also confirmed by Figure 4-4, which gives the average absolute errors in predicting the bubble compositions.

A careful observation of Figure 4-3, which gives the regression R-values in predicting the bubble pressures, further clarifies the selection. Based on this figure using seven hidden neurons is better than using six hidden neurons. The average absolute errors in predicting the bubble pressures, shown in Figure 4-2 clearly indicate the optimality of using seven neurons in the hidden layer.

So, the 2-7-2 neural network topology was taken as the optimum topology. A topology with the smaller number of hidden neurons and with good performance is the one we are looking for. An MLP with too many neurons memorize each exemplar pair (noise and all) rather than a general form in the same way as a polynomial of high degree memorizes each data points [12].

Further analysis on the performance of the proposed model is usually essential. The program EXTRACT.m is used to do such analysis. If the proposed model gives good result under different measures then the parameters of this topology is extracted and put in a separate file. Results obtained for Ethanol-Benzene are shown in figures bellow.

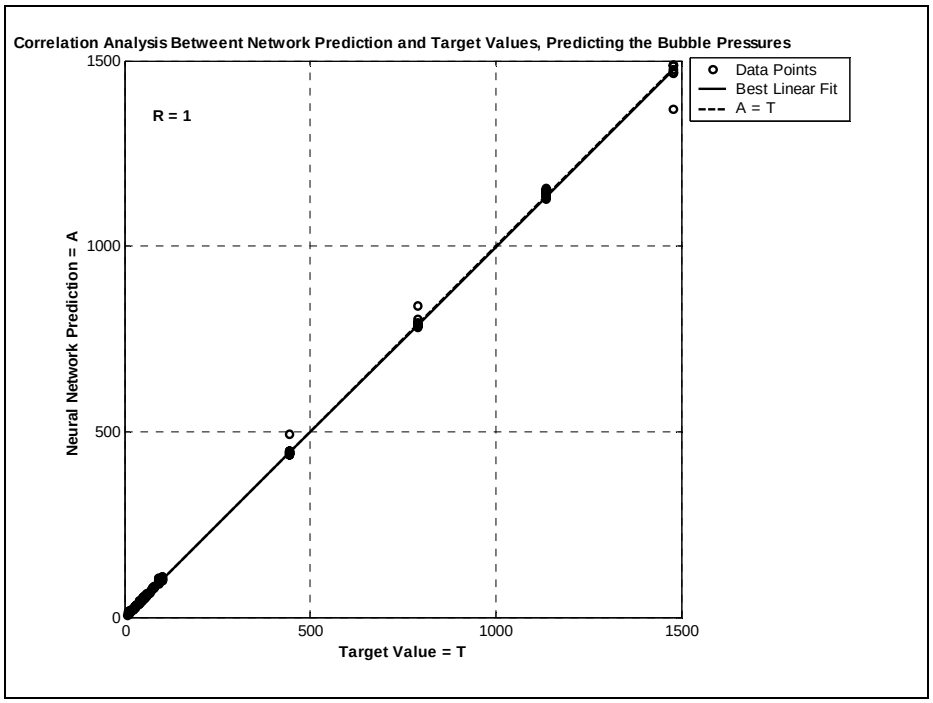

Figure 4-6 Correlation analysis for the proposed network topology, *Bubble-P calculation, predicting bubble-point Pressures*

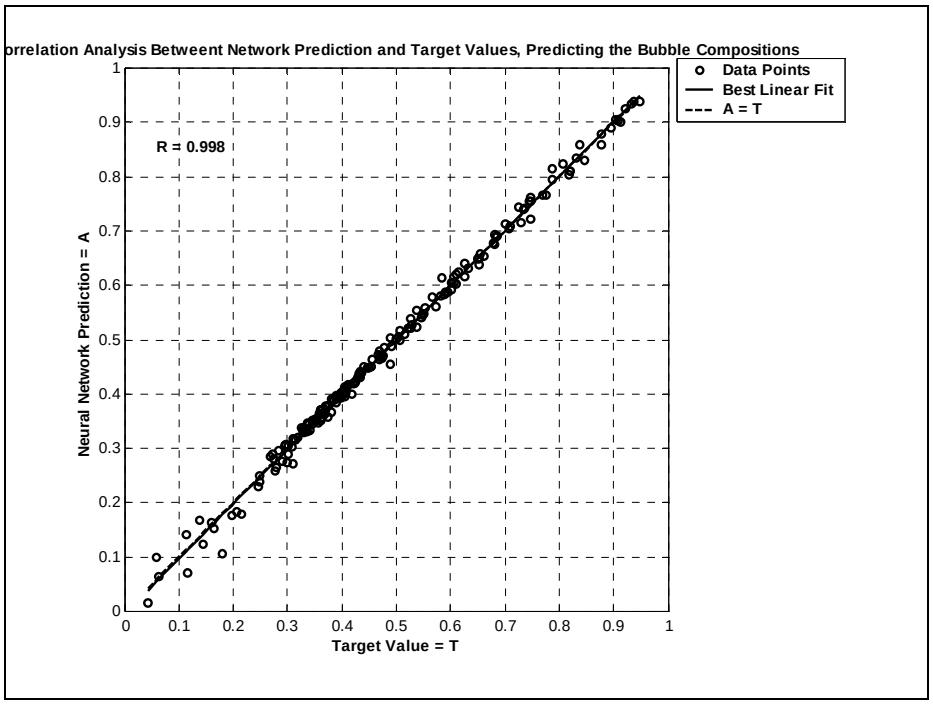

Figure 4-7 Correlation analysis for the proposed network topology, *Bubble-P calculation, predicting bubble compositions*

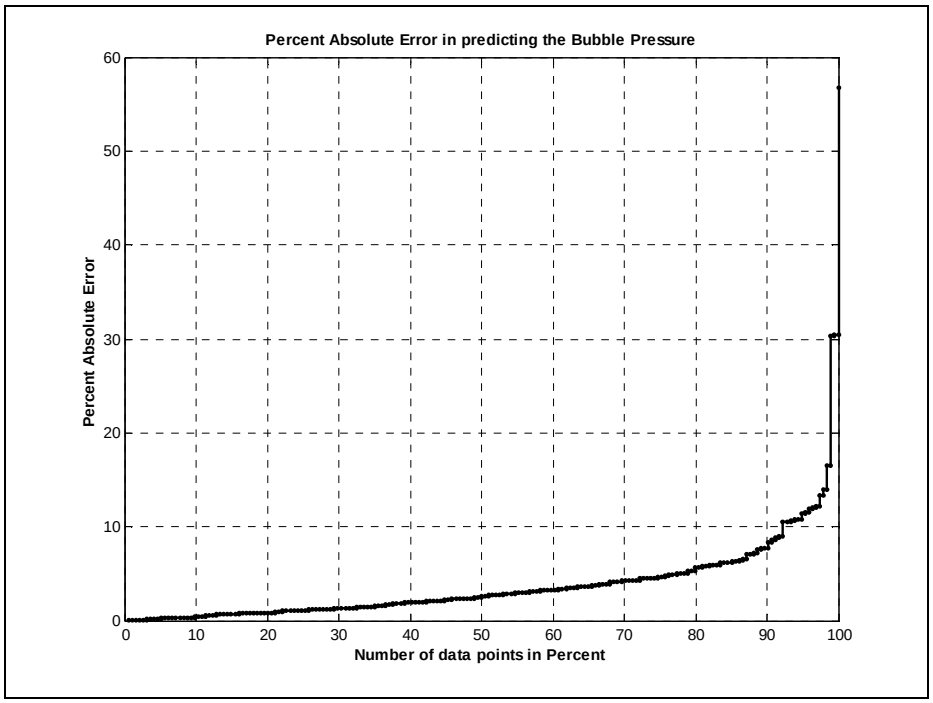

Figure 4-8 Percent absolute error distribution, *Bubble-P calculation, predicting bubble-point pressures*

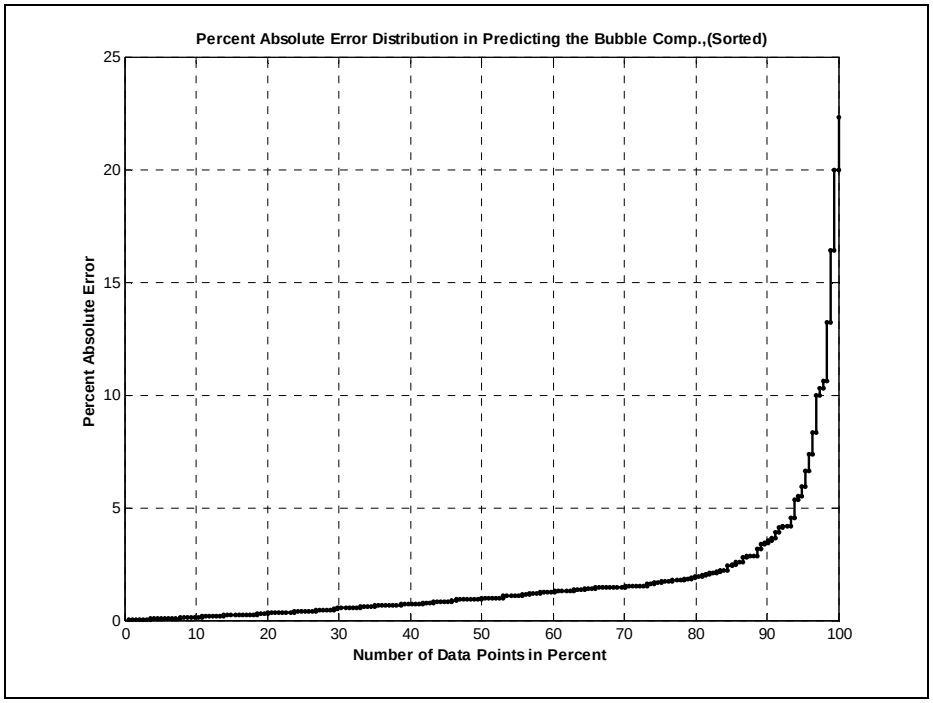

Figure 4-9 Percent Absolute error distribution, *Bubble-P calculation, predicting bubble compositions*

As it can be seen from the above figures the results obtained by using the proposed 2-7-2 topology are satisfactory. The regression R-value in predicting the bubble pressure is almost unity and 0.998 in predicting the bubble compositions. Figure 4-8, which shows the percent absolute error distribution in predicting the bubble-point pressure, is a good visual evaluator of the performance of the model. 95% of the total bubble pressure predictions were with a percent absolute error of less than 11.5 %. The average absolute error in using the proposed neural network model to predict the bubble-point pressures is 3.9148 Kpa.

From Figure 4-9 we can see that 95% of predicting bubble compositions were with a percent absolute error less than 6 %. The average absolute error in using the proposed neural network model to predict the bubble compositions was 0.0087.

Based on the above analysis the proposed network topology gives excellent results. So, the value of the synaptic weights and biases of this neural network topology are extracted and saved to construct the model in a form of mathematical function CONSTRUCT.m is the MATLAB function file which put the the proposed neural network model in its operating mode.

Program USE.m is the MATLAB script file which call CONSTRUCT.m, the constructed neural network model, to do the required analytical work. For instance, USE.m can be designed to calculate the P-x-y diagram as shown in Figure 4-10 for three equilibrium temperatures.

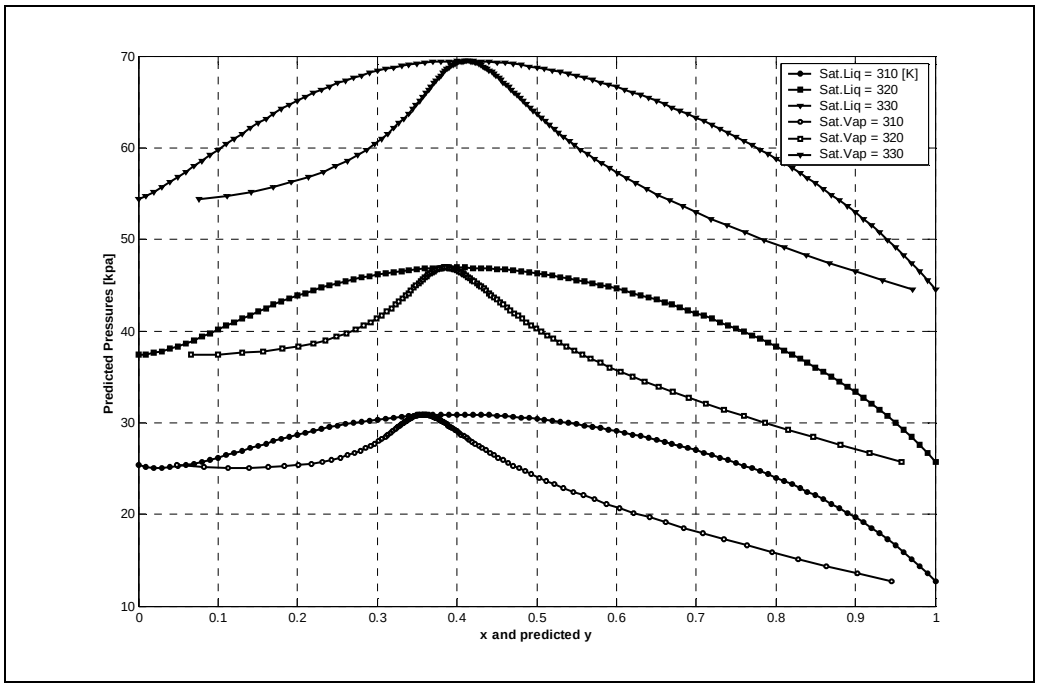

Figure 4-10 Calculated P-x-y diagram, *using the 2-7-2 topology, Ethanol-Benzene*

The proposed neural network model clearly shows the thermodynamically expected P-x-y diagram. The ease of getting such VLE diagram from a single model is really praiseworthy. The inclusion of as many data as possible in the training and testing phase is critical. The data collected should cover the temperatures and pressures ranges in a smooth fashion. The problem of insufficient data during network construction activities were faced in the case of some binary mixtures studied in the thesis.

For instance, the Bubble-P type neural network model developed for *Ethanol-Water*, *Methanol-Water,* and *Methyl acetate-Water* show great discrepancy from the expected fact. The reason for this is the data sets used for the network training. The VLE data obtained for these binary mixtures do not cover the pressure range smoothly. Different data transformation methods were also tried, but they gave no improvement. So, collecting set of data which cover the pressure range smoothly is critical.

4.2.1.2 Bubble-T calculation

The bubble-T calculations also pursue the same procedure as the bubble-P calculations shown above. The graphical representation of the model is shown in Figure 4-1 (b)

Using sigmoid type activation function in the hidden layer and linear type in the output layer is the better choice. The Levenburg-Marguardt type training algorithm gives a faster training and better performance as in the case of the bubble-p calculation.

Graphical outputs showing the performance of neural networks with different number of neurons in their hidden layer is used to select the optimum neural network topology. This result is shown in Figure 4-11 and Figure 4-12 below.

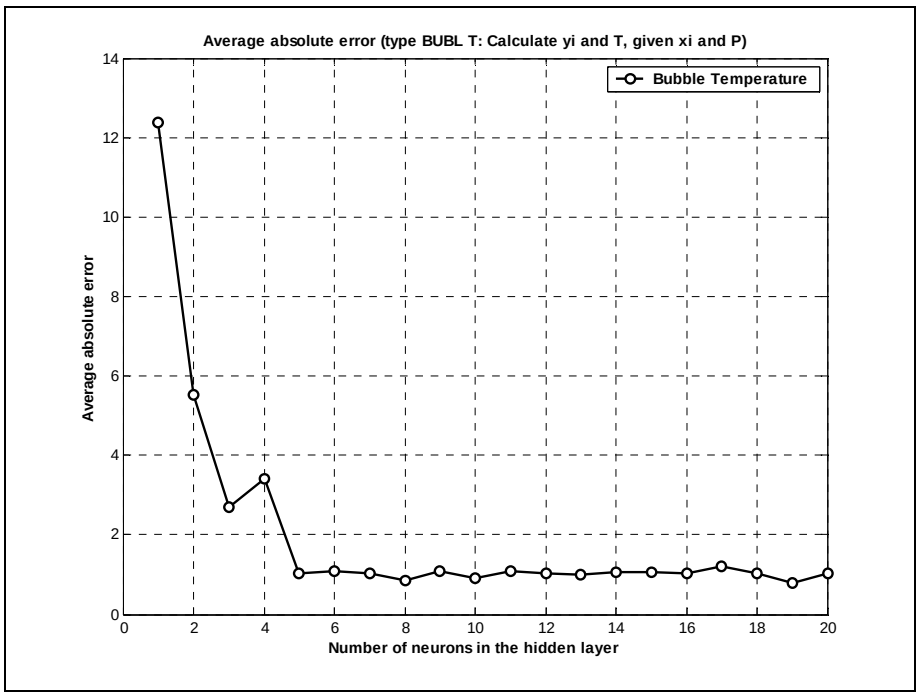

Figure 4-11 Average absolute error based performance measures, *Bubble-T calculation, predicting bubble-point temperature, Ethanol _Benzene*

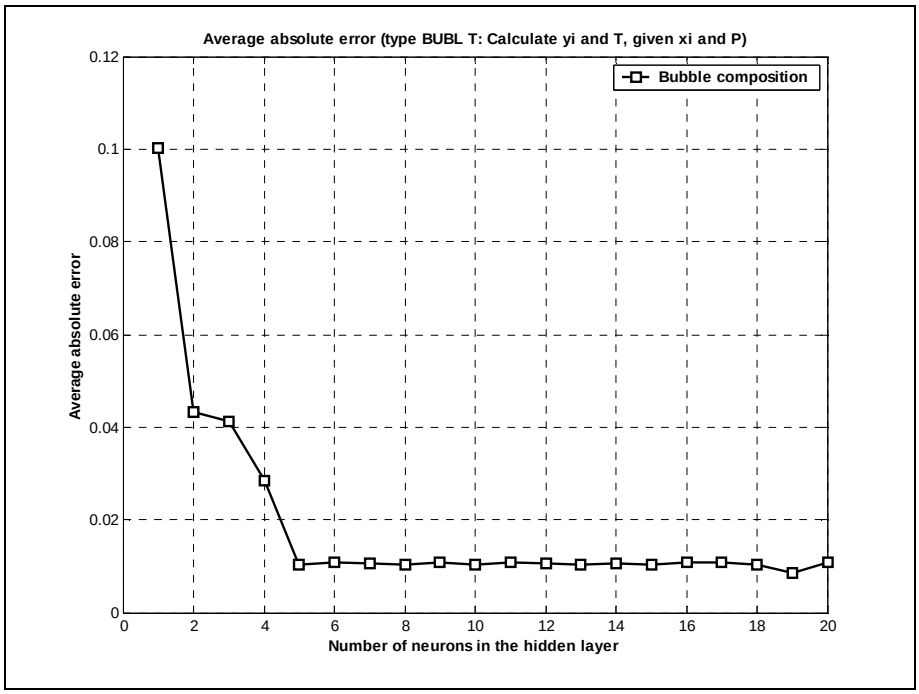

Figure 4-12 Average absolute error based performance measures, *Bubble-T calculation, predicting the bubble composition, Ethanol _Benzene*

The analysis on the performance of the different neural network topologies in predicting the bubble compositions and the equilibrium temperatures using the liquid compositions and the equilibrium pressures as input to the network is shown above. The decision of selecting the best neural network topology was made by a careful analysis of the figures as done in the case of bubble-P calculation. Doing such analysis shows the 2-7-2 neural network as the optimum topology. Further analysis on the performance of the proposed neural network model was made and results are shown in figures below.

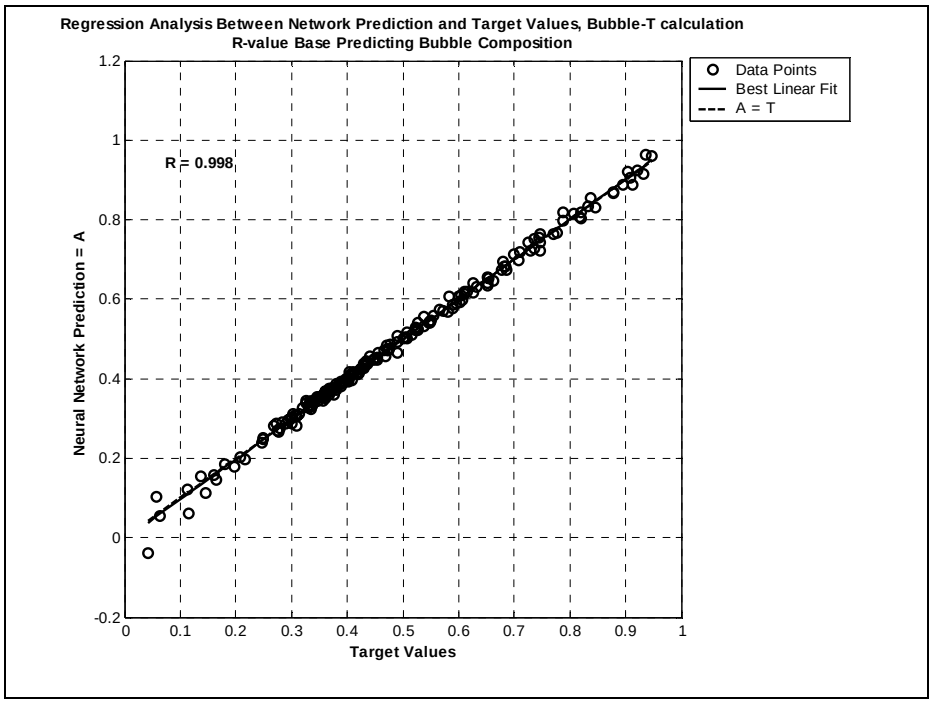

Figure 4-13 Correlation analysis for the proposed network topology, *Bubble-T calculation, predicting bubble compositions*

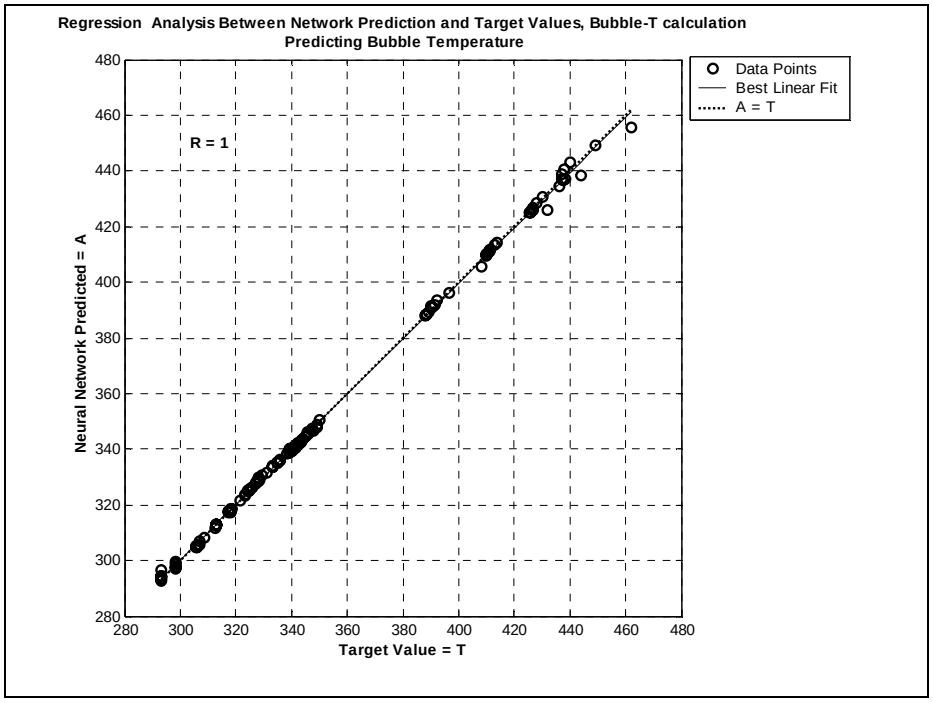

Figure 4-14 Correlation analysis for the proposed network topology, *Bubble-T calculation, predicting bubble-point temperature*

The correlation between network prediction and target values using the proposed 2-7-2 networks topology gives a good result as can be seen from Figure 4-13 and 4-14 in predicting the bubble compositions and bubble temperatures, respectively. The average absolute errors in using this topology to predict the bubble compositions and the bubble temperatures are 0.0085 and 0.652 ^{o}C respectively. 95% of the bubble temperatures were predicted with a percent absolute error of less than 0.442 % and 95% of the bubble compositions were predicted with a percent absolute error of less than 7.4%

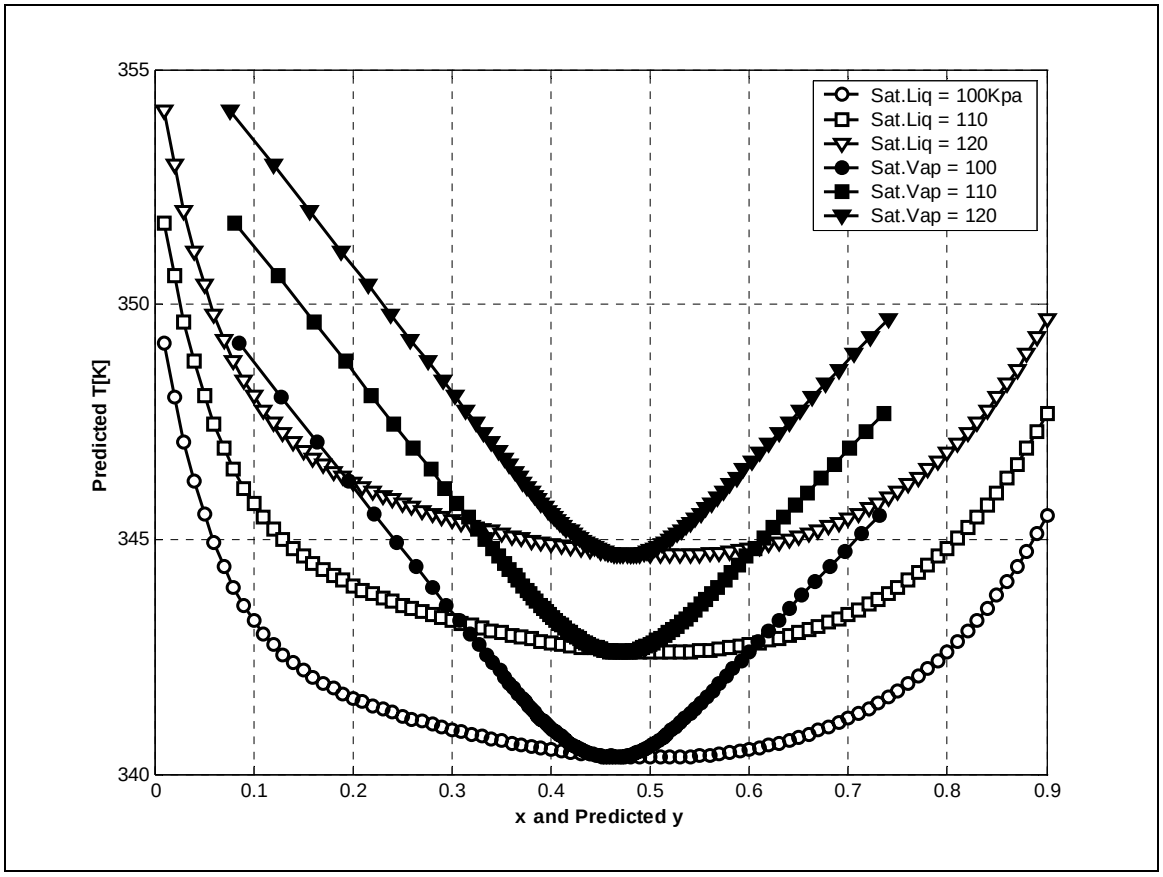

Figure 4-15 Calculated T-x-y diagram for *Ethanol-Benzene using the Bubble-T neural network model*

4.2.1.3 Dew-P calculations

The same procedure is followed to construct a neural network model for the dew-p calculation. The graphical interpretation of the model is shown in Figure 4-1 (c). In this model the equilibrium dew pressures and liquid phase compositions are predicted using the equilibrium temperature and the vapor phase composition as input to the model.

The dew-P calculation is ok for the case of Ethanol-Benzene as shown in figures below. The optimum network topology in dew-p calculation is the 2-8-2.

But, the same is not true for some of the sited binary mixtures. The difficulty of covering the pressure range in a smooth manner in the collected VLE data is the cause of the discrepancy. This problem again show the criticality of the nature of data used in neural network training.

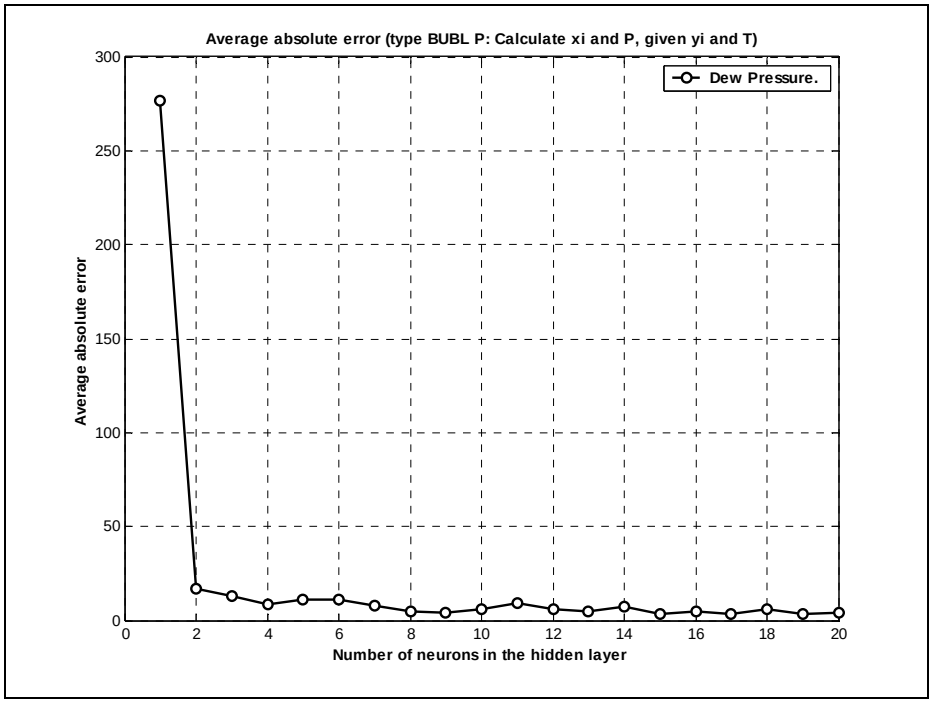

Figure 4-16 Average absolute error based performance measures, *Dew-P calculation, Predicting the dew-point pressure, Ethanol _Benzene,*

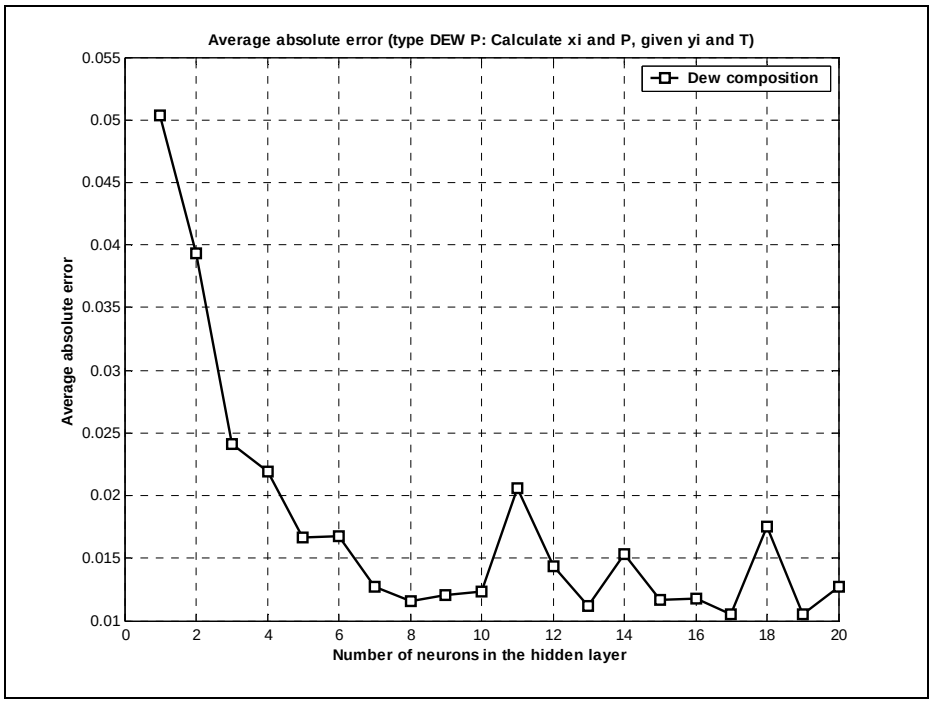

Figure 4-17 Average absolute error based performance measures, *Dew-P calculation Predicting the dew composition, Ethanol _Benzene,*

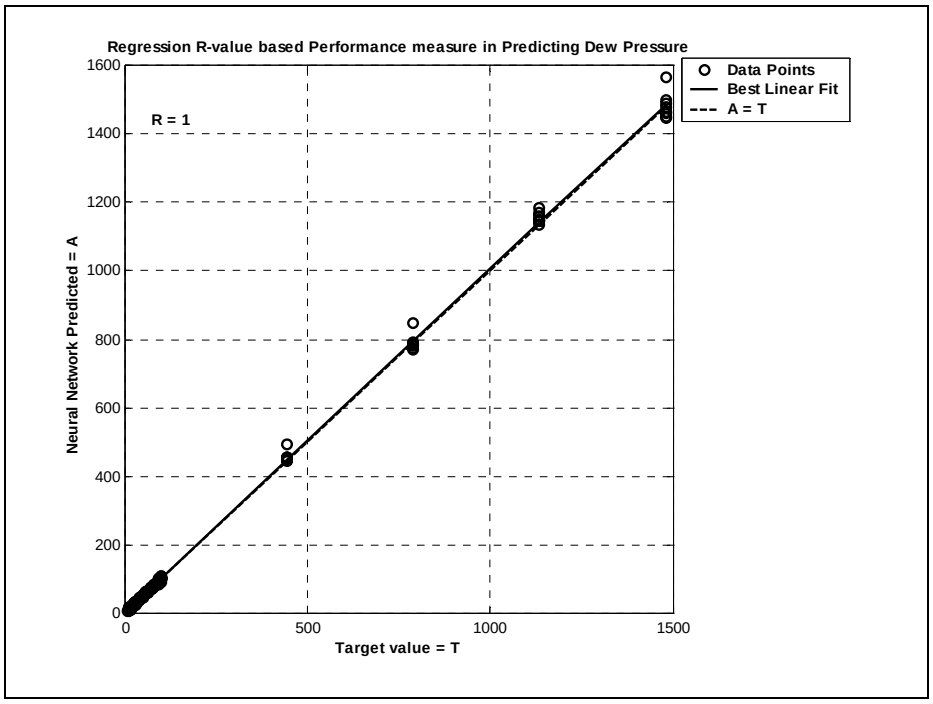

Figure 4-18 Correlation analysis for the proposed topology, *Dew-P calculation, predicting dew-point Pressure, Ethanol _Benzene*

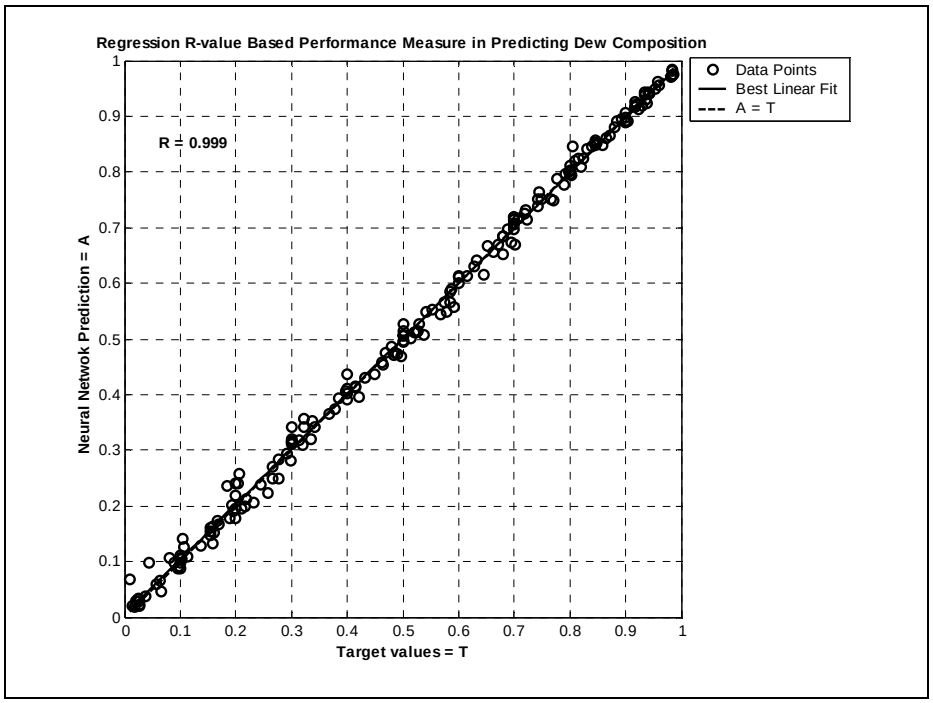

Figure 4-19 Correlation analysis for the proposed topology, *Dew-P calculation, predicting dew composition, Ethanol _Benzene*

The correlation between network prediction and target values using the proposed 2-8-2 network topology gives a good result as can be seen from figure 4-18 and 4-19. The average absolute errors in using this topology to predict the dew compositions and the dew pressures were 0.0116 and 5.0784 Kpa, respectively.

4.2.1.4 Dew-T calculations

The Dew-T calculation also follows similar procedure. For most of the binary mixtures encountered in the study the result obtained for the dew-T calculation is excellent. The graphical interpretation of this neural network model is shown in Figure 4-1 (d). As it can be seen from Figure 4-20 and 4-21 the optimum network topology in dew-T calculation is the 2-7-2. Further analysis on the selected network topology was done as usual and results are shown in figure 4-22 and 4-23.

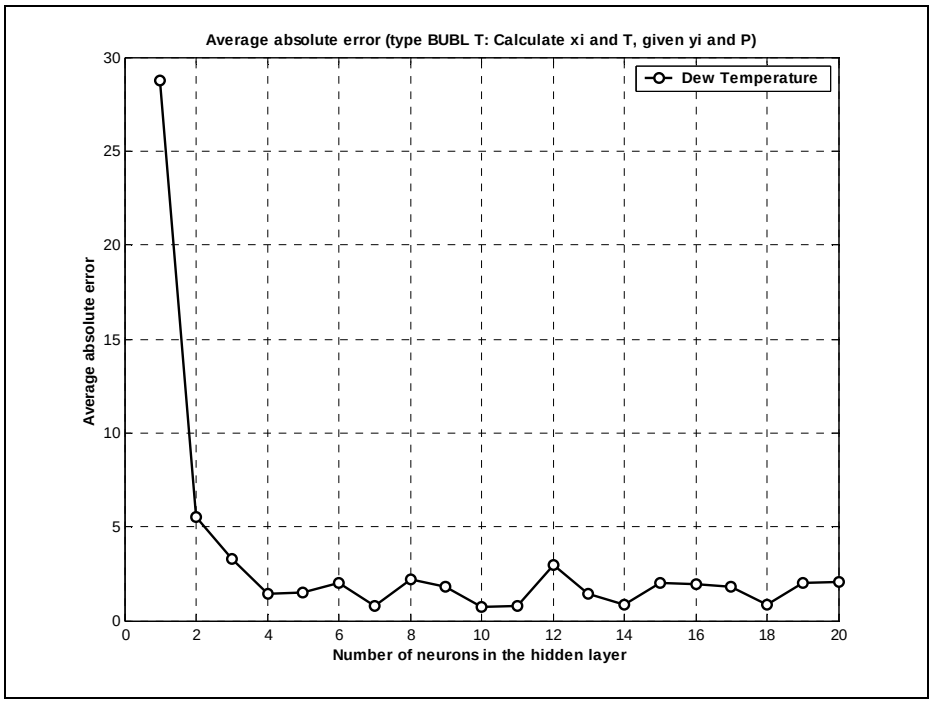

Figure 4-20 Average absolute error based performance measures, *Dew-T calculation, predicting the dew-point temperature, Ethanol _Benzene*

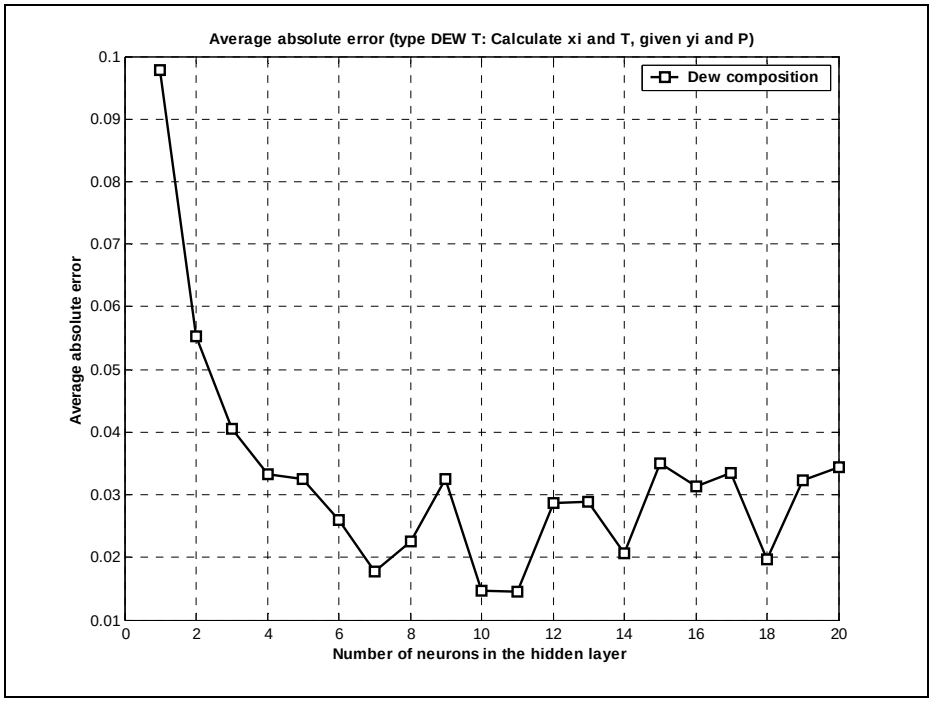

Figure 4-21 Average absolute error based performance measures, *Dew-T calculation, predicting the dew composition, Ethanol _Benzene,*

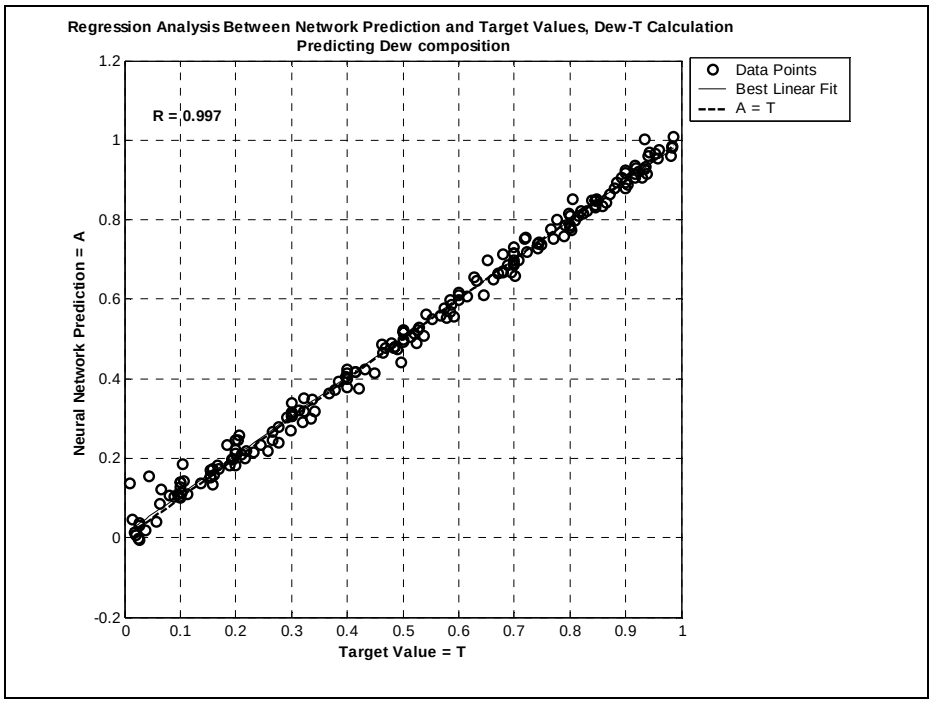

Figure 4-22 Correlation analysis for the proposed network topology, *Dew-T calculation, predicting dew composition*

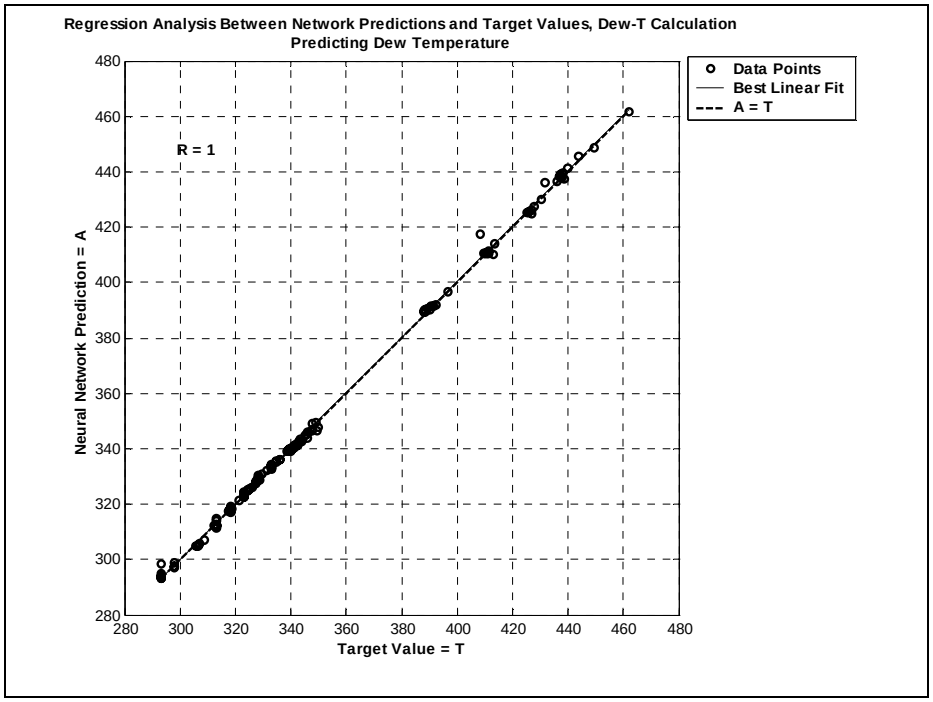

Figure 4-23 Correlation analysis for the proposed network topology, *Dew-T calculation, predicting dew-point temperatures*

The correlation between network prediction and target values using the proposed 2-7-2 networks topology gives a good result as can be seen from Figure 4-22 and Figure 4-23. The average absolute errors in using this topology to predict the dew compositions and the dew-point temperatures were 0.0177 and 0.7671 ^{o}C , respectively.

The proposed model was used to calculate the T-x-y diagram and the result is shown in Figure 4-24

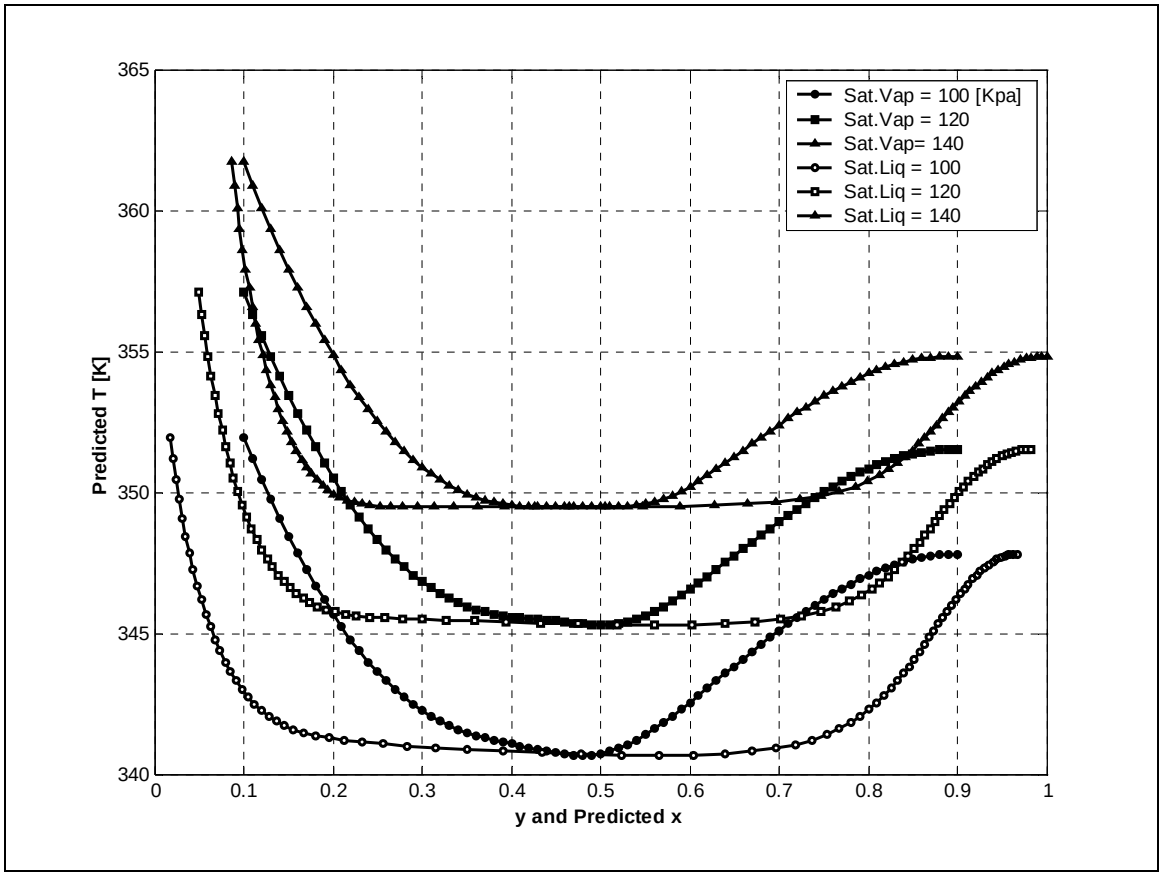

Figure 4-24 Calculated T-x-y diagram for *Ethanol-Benzene using the Dew-T neural network model*

4.2.1.5 Summary of results obtained

Due to space limitation, it is not possible to show the result obtained for all the binary mixtures modeled in this thesis work. Appendix 6.1 summarizes the result obtained. There are totally 12 separate binary mixtures considered in this thesis. Each binary mixture needs four separate neural network model to address the four practically valuable problems as shown above. So, the total number of neural network models developed equals 48.

The result obtained in Bubble-T and Dew-T calculations were excellent but the Bubble-P and Dew-P calculations show relatively lower performance for the majority of the binary mixtures sited in this thesis. The cause of this discrepancy is in the nature of the collected data. Most of the collected VLE data do not cover the pressure range in a smooth fashion and this creates conditions for the poor performance encountered. The result of the study confirms the criticality of data collection. Collected data for neural network training should cover the range required in a smooth fashion so that the network model picks out the input-output relationship in a correct manner.

The T-x-y and P-x-y diagrams obtained by using the neural network models as shown for the Ethanol-Benzene in the above discussion are good indicators of the generality of the models showing their thermodynamic consistency.

The performance of the Bubble-P and the Dew-P calculations can be improved by collecting a good set of data, covering the pressure ranges in a smooth fashion. For instance, by taking portion of the VLE data where pressure ranges are smoothly covered show reasonable improvement on the performance of some of the models. As to training a network on a single constant pressure or constant temperature VLE data, there is no problem. The relatively lower performance of the models is not due to the incapability of the neural network paradigm, it is rather due to the data used in the training steps.

Neural network models train on collection of VLE data at different equilibrium conditions simultaneously and gives a single VLE prediction model as shown for the twelve binary mixtures in this thesis. In using the conventional thermodynamic based numerical methods we need to determine the equation parameters for a given system at a given constant temperature or constant pressure VLE data. There are a lot of iteration steps and there is additional problem of selecting the appropriate thermodynamic model for the system in consideration. Comparing the neural network models with the corresponding thermodynamic based numerical models like Van Laar and Margules models will be discussed in the next section.

4.2.1.6 Comparing ANN models with thermodynamic models

Table 4-1 compares the prediction performance of the neural network based VLE prediction model developed for Ethanol-Benzene in the previous discussion with two of widely used thermodynamic based numerical VLE prediction models. The comparison is based on a single constant temperature VLE data, and further analysis may be needed at various conditions to give a general conclusion. The table depicted the fact that, a neural network model trained on various constant pressure and constant temperature data give a better performance than the two conventional VLE calculation methods- Van Laar and Margules, at least for this particular condition.

Table 4-1 Comparison of NN with thermodynamic method

COMPARISON OF THE NEURAL NETWORK PREDICTION WITH CONVENTIONAL VLE CALCULATION METHODS							
Exp. Values					Bubble Y prediction		
x	T,k	y	P[Kpa]		NN	Margules	Van Laar
0.020	313.000	0.145	27.786		0.138	0.109	0.119
0.095	313.000	0.280	31.972		0.276	0.289	0.306
0.204	313.000	0.332	33.212		0.325	0.335	0.375
0.378	313.000	0.362	33.639		0.364	0.367	0.384
0.490	313.000	0.384	33.172		0.389	0.371	0.385
0.592	313.000	0.405	32.759		0.416	0.387	0.395
0.702	313.000	0.440	31.639		0.453	0.428	0.424
0.802	313.000	0.507	29.252		0.511	0.506	0.483
0.880	313.000	0.605	25.772		0.603	0.616	0.575
0.943	313.000	0.747	22.599		0.749	0.768	0.721
0.987	313.000	0.912	19.413		0.901	0.935	0.909
				R^2 =	0.999	0.998	0.988

As can be seen from the above table the neural network model give a better result. Artificial neural networks, a purely data driven modeling technique, give a mathematical model of the process using only observed data with good performance. Designing a mathematical model of a process using only observed data is an advantage and further analysis to this technique in a well organized and profound manner is really a big prospect to consider.

4.2.2 VLE prediction model for group of binary mixtures

This section attempts to construct a generalized neural network model for VLE calculation of a group of binary mixtures. VLE data for the twelve binary mixtures collected previously were used here. For this analysis a constant pressure VLE data at 101.33Kpa of each binary mixture were integrated together to give a single data set to be used in the network construction.

Easily obtainable pure component properties were gathered and used as additional input parameters. The selected pure component properties are

- Molecular weight of the component , M
- Critical Temperature, T_C
- Critical Pressure, P_C
- Critical volume, V_C
- Normal boiling temperature, T_N
- Accentric factor, ω
- Compressibility factor at critical point, Z_C

These pure component properties together with the P-x-y-T VLE data for each binary mixture are used as input output mapping. So, the data arrangement gives a 15 dimensional input vector and a two dimensional output vector as shown in figure below.

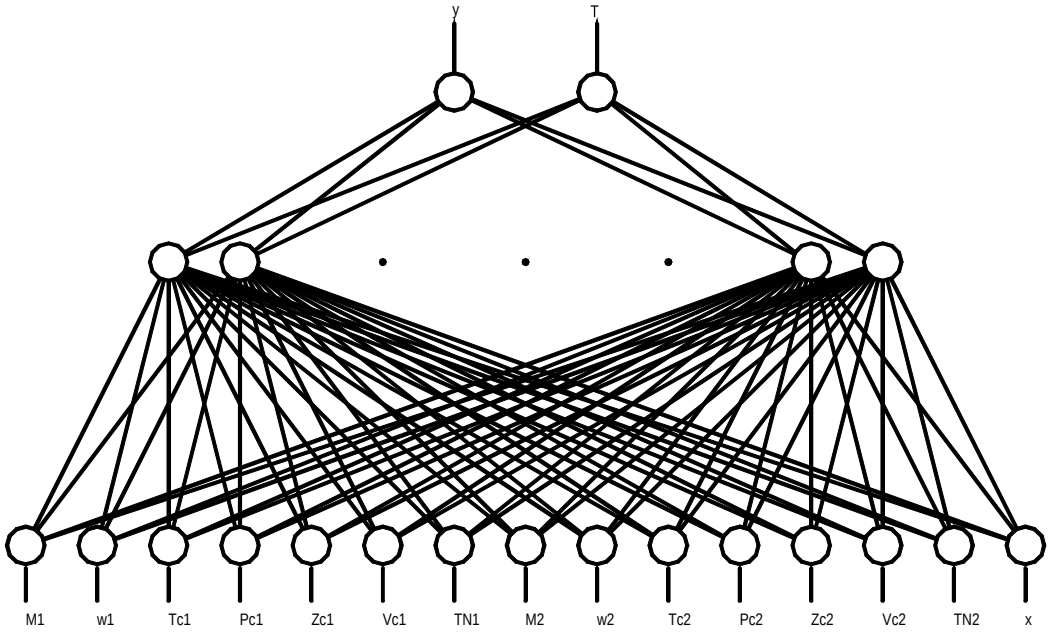

Figure 4-25 Neural network model to predict the bubble point

4.2.2.1 Prediction model without feature extraction

Integrating the collected VLE data of the twelve binary mixtures with their corresponding pure component properties in a format to fit the model shown in Figure 4-25 is the first step. Data normalization step using the linear transformation method is applied. Using the developed MATLAB program COMPARE.m to construct a set of competing neural network models, neural network models with different number of neurons in their hidden layer were analyzed. Results are shown bellow

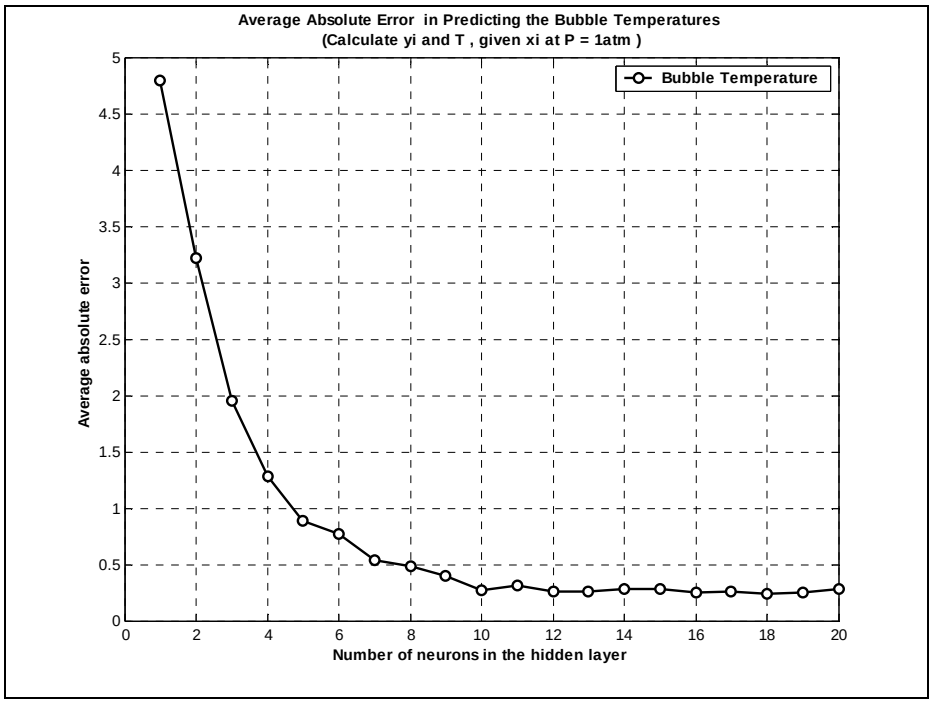

Figure 4-26 Average absolute error based performance measures, *group of binary mixtures, predicting bubble-point temperatures, network model based on all collected features*

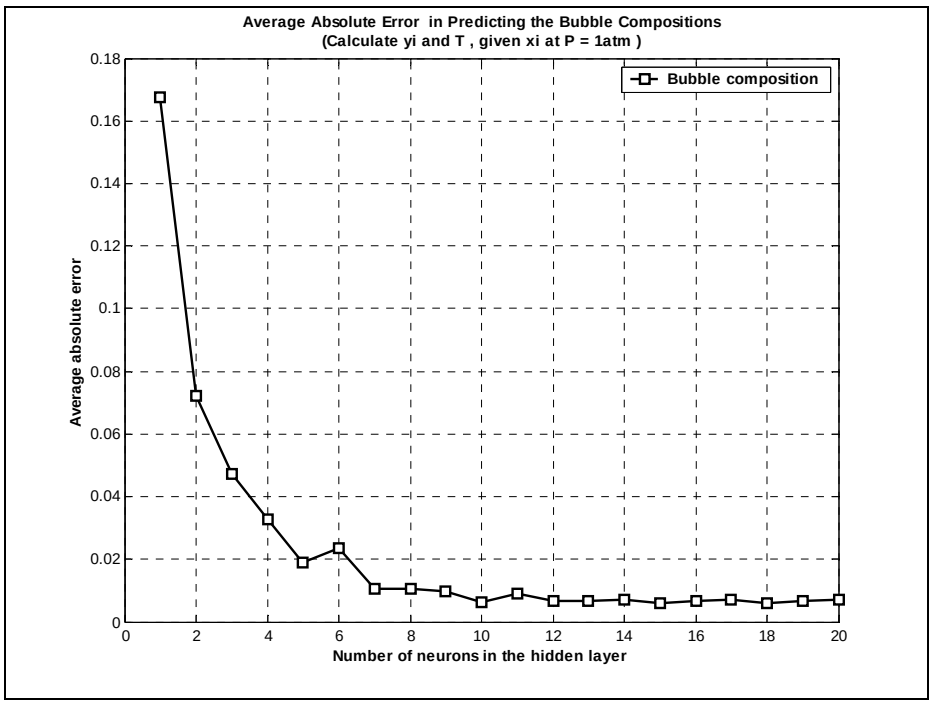

Figure 4-27 Average absolute error based performance measures, *group of binary mixtures, predicting bubble compositions, network model based on all collected features*

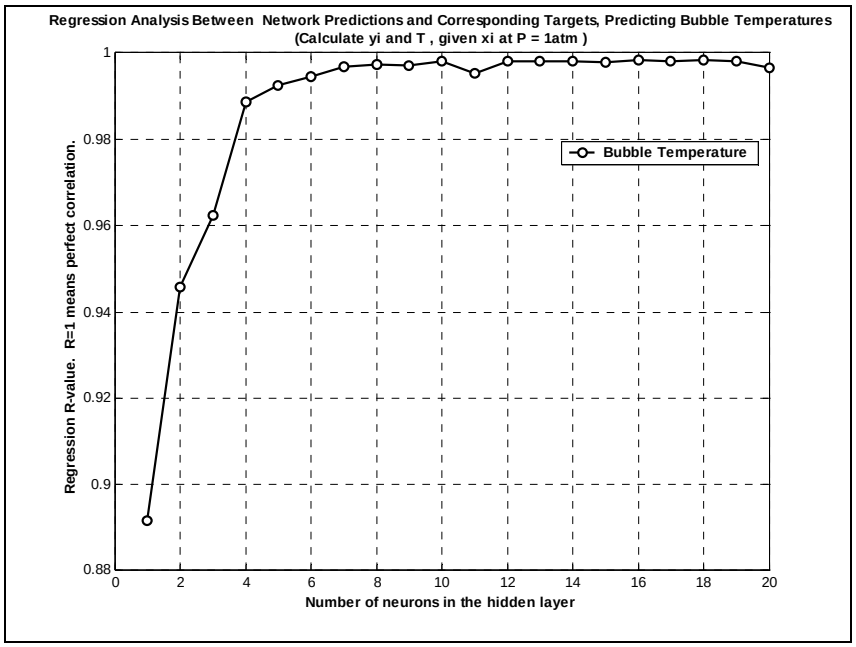

Figure 4-28 Regression R-value based performance measures, *group of binary mixtures, predicting bubble point temperatures, network model based on all collected features*

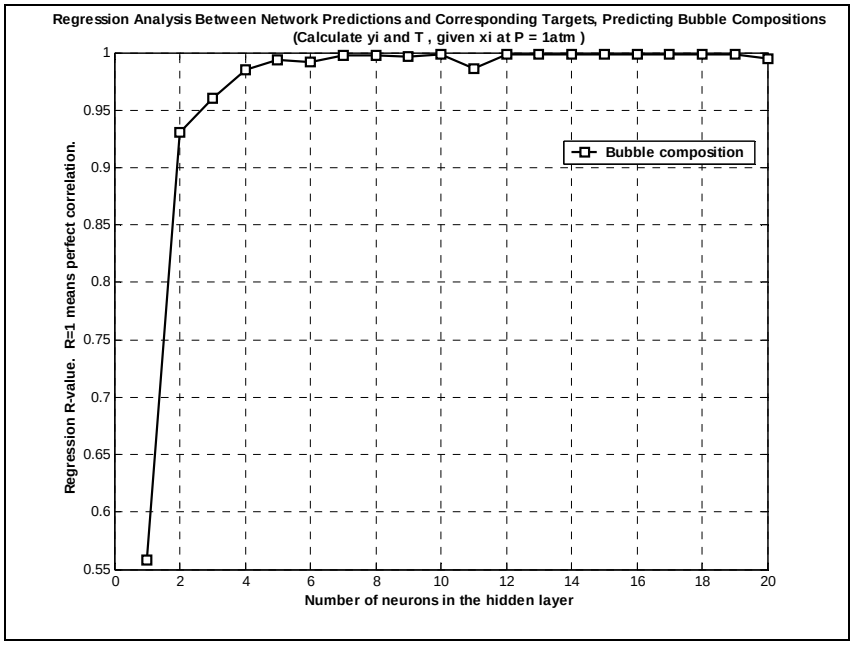

Figure 4-29 Regression R-value based performance measures, *group of binary mixtures,, predicting bubble compositions, network model based on all collected features*

The average absolute error and the regression R-value based performance measures shown in the above four figures suggested a network topology of 15-10-2 as the optimum. Further analyses on the performance of this model are shown in figures below.

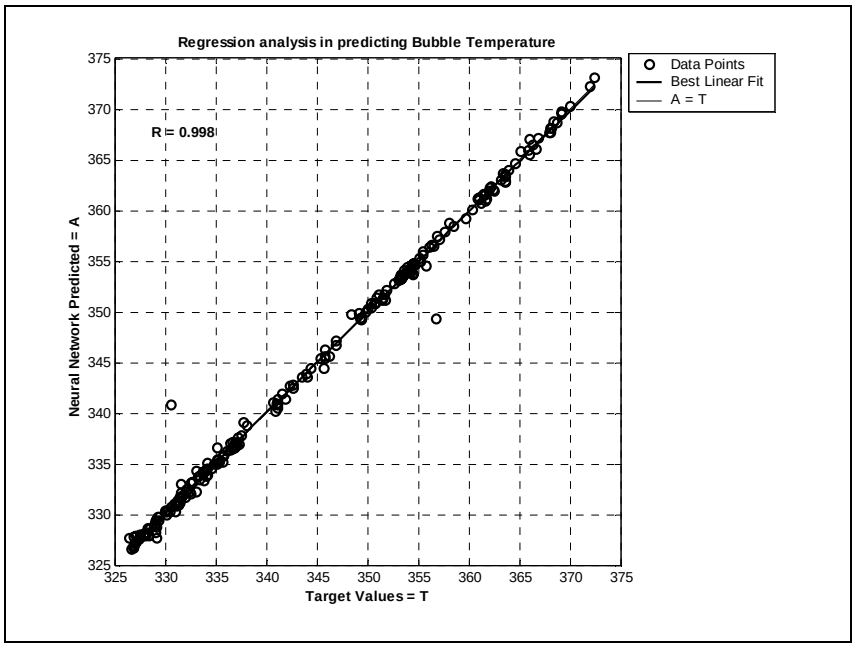

Figure 4-30 Correlation between the network prediction and the target values for the proposed network topology, *group of binary mixtures, predicting bubble temperatures*

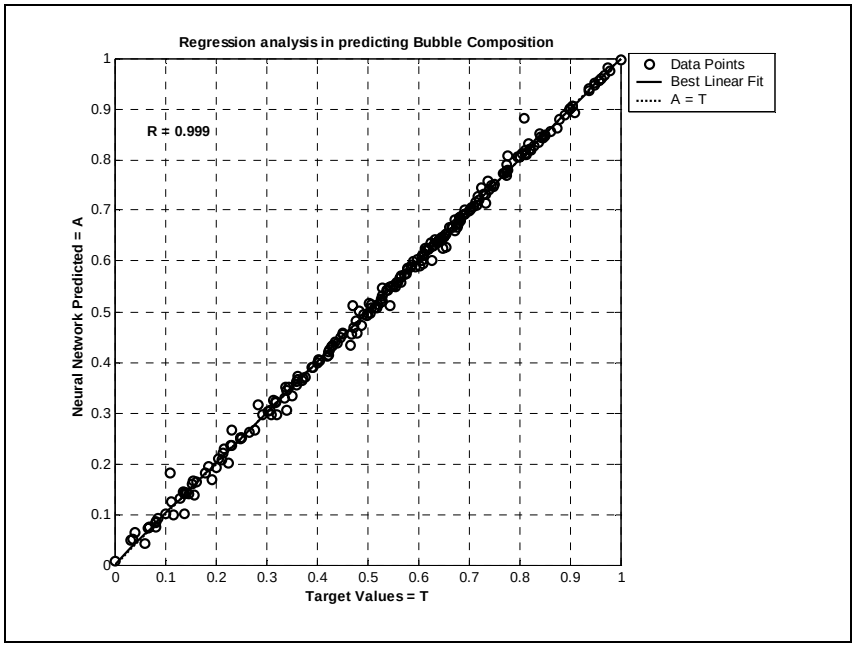

Figure 4-31 Correlation between the network prediction and the target values, for the proposed network topology, *group of binary mixtures, predicting bubble compositions*

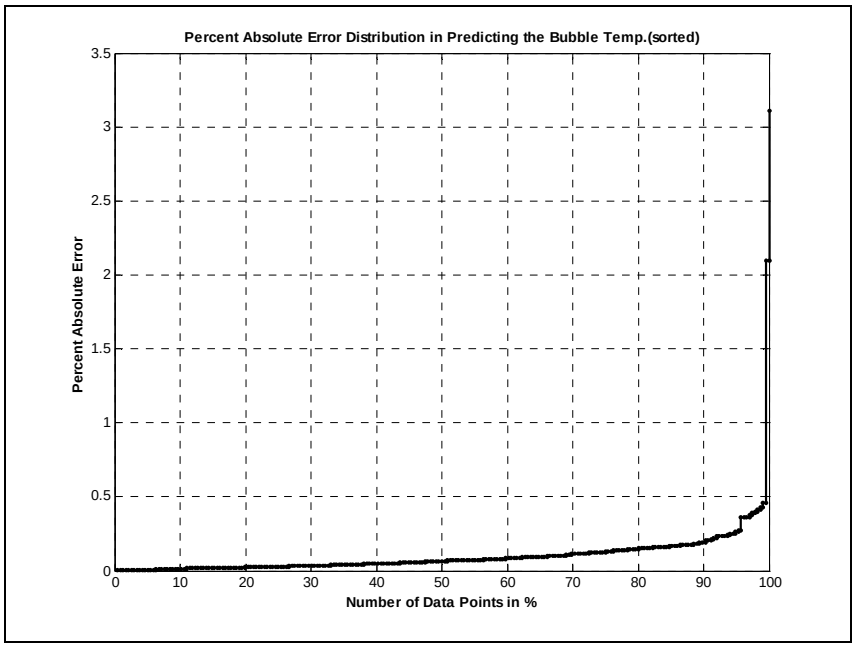

Figure 4-32 Percent absolute error distribution, *predicting the bubble point temperatures using the proposed neural network model, group of binary mixtures*

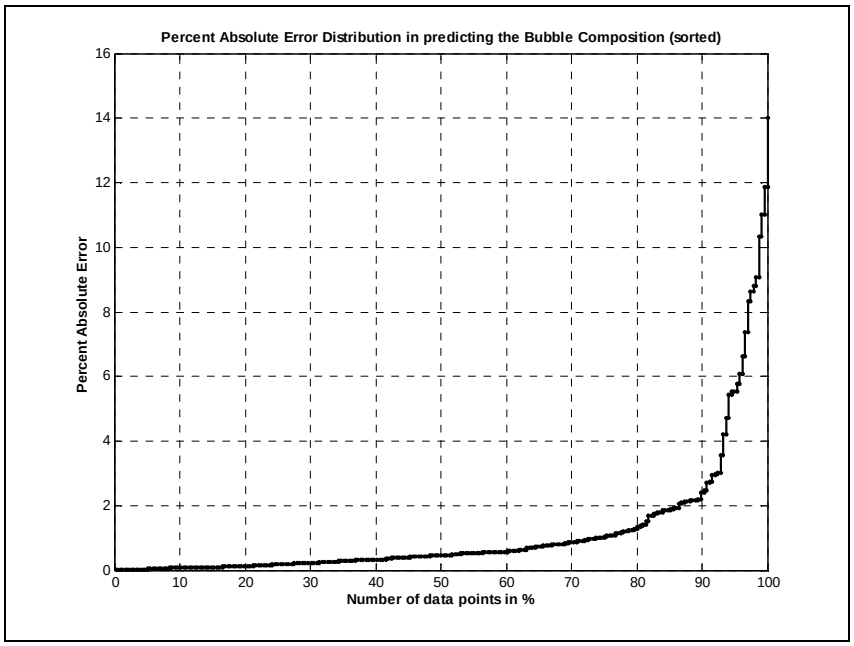

Figure 4-33 Percent Absolute error distribution, *predicting the bubble compositions using the proposed neural network model, group of binary mixture*

The regression R-value in predicting the bubble temperature and the bubble compositions are equal to 0.998 and 0.999 respectively. The distribution of the percent absolute errors in predicting bubble-point temperatures is shown in Figure 4-32. From this figure we can

see that 99% of the total predictions were made with a percent absolute error less than 0.46%. The percent absolute error distribution in predicting the bubble compositions were shown in Figure 4-33. The maximum percent absolute error is equal to 14% and 95% of the predictions were within a percent absolute error of 5.53%.The average absolute error in predicting the bubble temperatures and the bubble compositions are equal to 0.2706 ^{o}C and 0.0065, respectively for temperature range of 326.45^{o}C to 372.45^{o}C.

4.2.2.2 Prediction model after feature extraction

The Belue-Bauer method of selecting features is applied on the VLE data which are collected for the above problem. The result of this feature extraction method is shown in Figure 4-34. Based on this figure, feature 2 and feature 6 which are properties of component one (i.e. the volatile component) and feature 11 and feature 14 which are the properties of the second component looks more significant over the others.

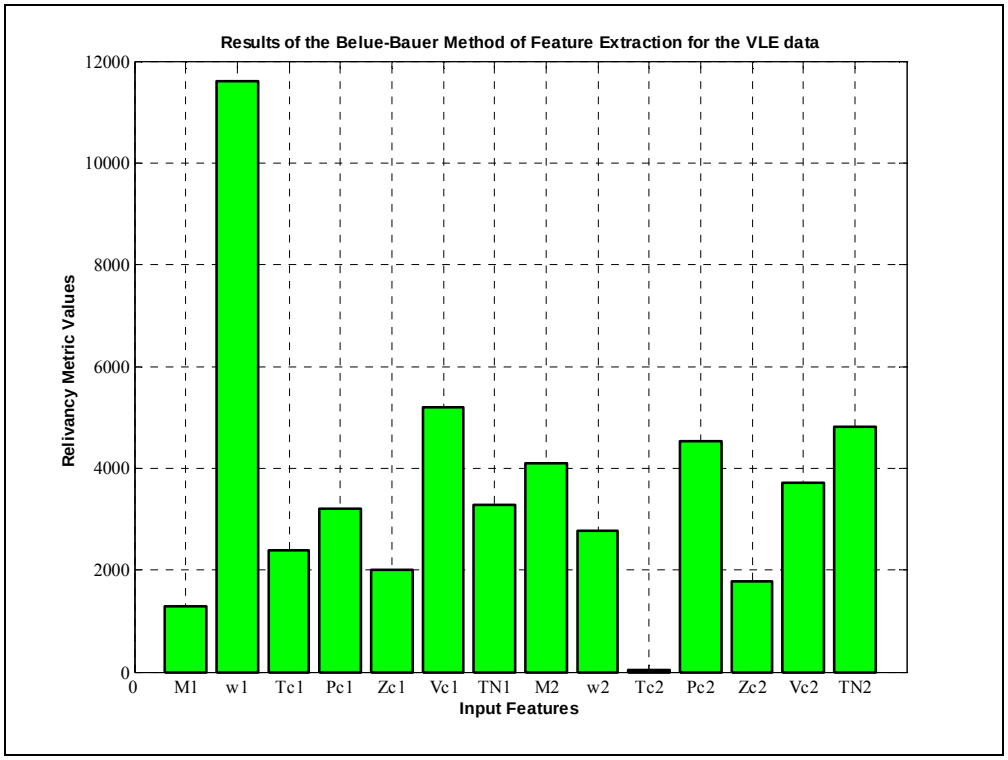

Figure 4-34 Results of the Belue-Bauer method of selecting features, *applied on the VLE data set for the group of binary mixtures*

Figure 4-34 suggests designing the neural network model using the following inputs:

- Feature 2, accentricity of the first component, **ω 1**
- Feature 6, critical volumes of the first component, **Vc1**
- Feature 11, critical pressures of the second component, **Pc2**
- Feature 14, Normal boiling temperature of the second component, $\mathbf{T_{N2}}$
- Feature 15, is the equilibrium liquid compositions, **x** where we want to predict their corresponding equilibrium vapor compositions and temperatures

Results obtained in designing a neural network model based on the extracted features are shown in figures below.

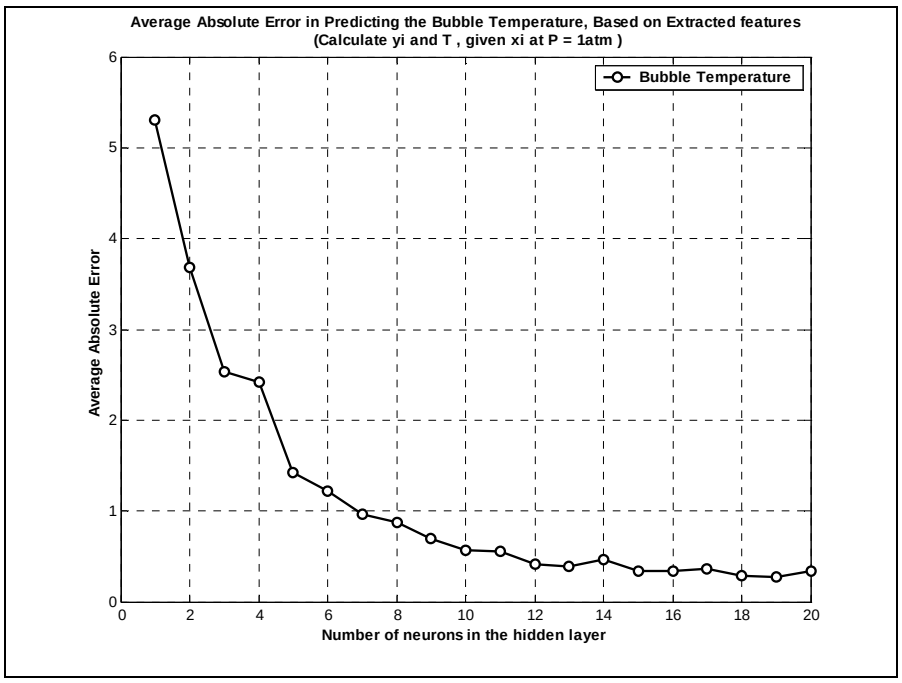

Figure 4-35 Average absolute error based performance measures after feature extraction, *predicting the bubble-point temperatures*

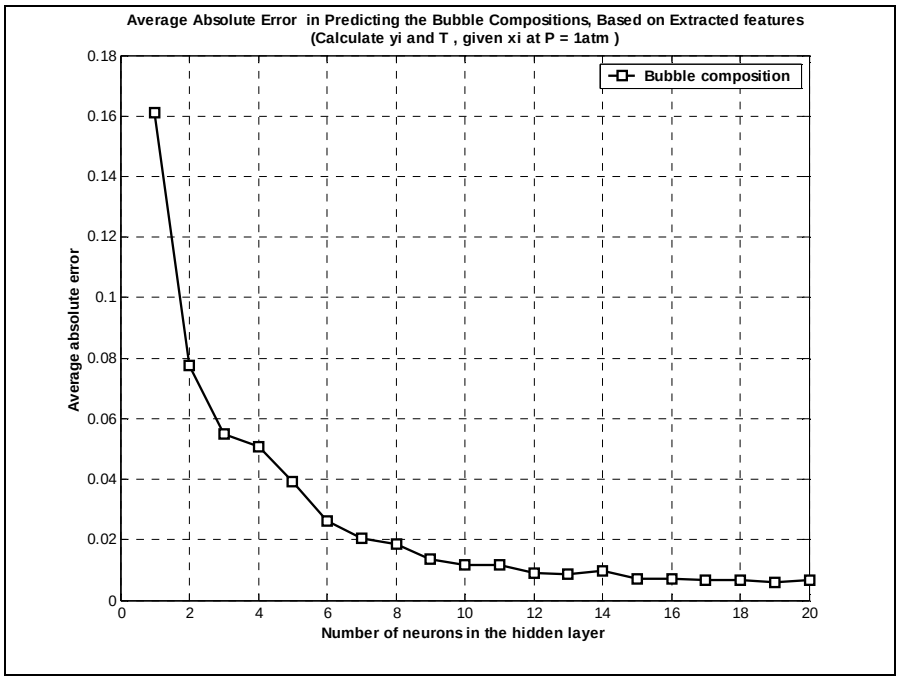

Figure 4-36 Average absolute error based performance measures after feature extraction, *predicting the bubble compositions*

The average absolute error based performance measures in predicting the bubble temperatures and the bubble compositions are shown in Figure 4-35 and Figure 4-36 respectively. These two figures suggested a 5-12-2 network topology as the optimum. Further analysis in using this network topology is assessed and results are shown in the following figures

The Regression R-value obtained in predicting the bubble temperatures and the bubble compositions show small decrement when compared to the result obtained for the model based on all feature vectors. The simplicity of the neural network developed by using the extracted features and the need for fewer pure component properties may make the performance decrease tolerable.

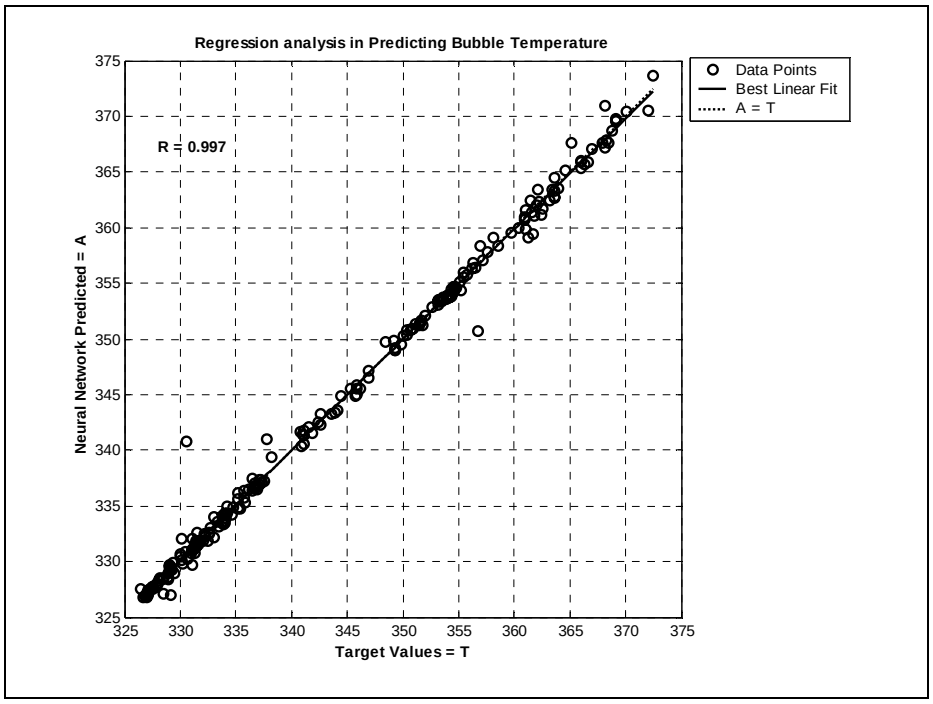

Figure 4-37 Correlation analysis for the proposed model, *after feature extraction, predicting the bubble-point temperatures*

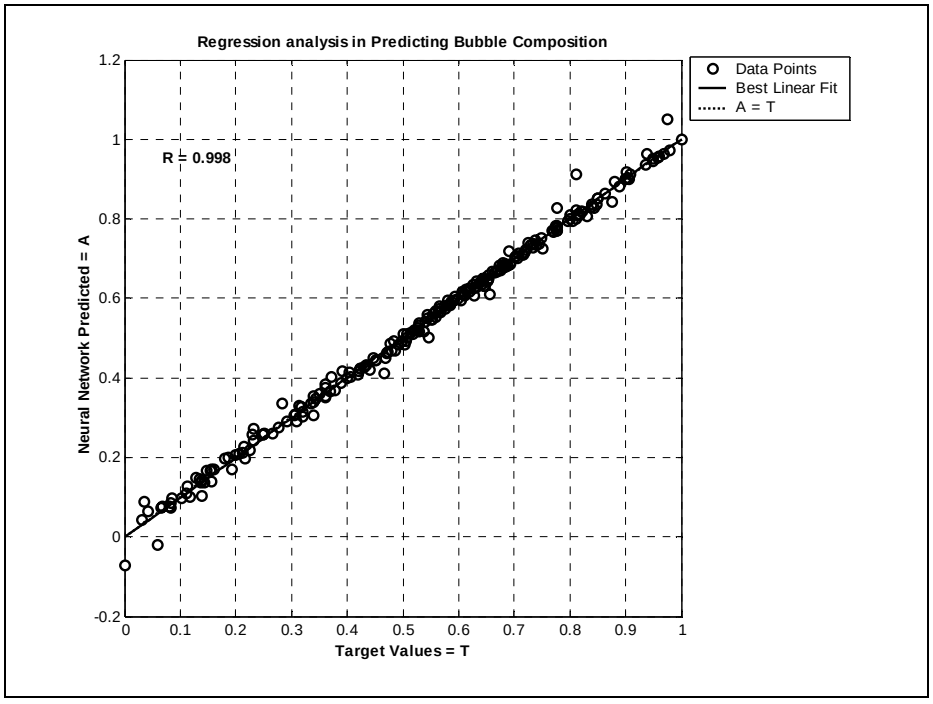

Figure 4-38 Correlation analysis for the proposed model, *after feature extraction, predicting the bubble compositions*

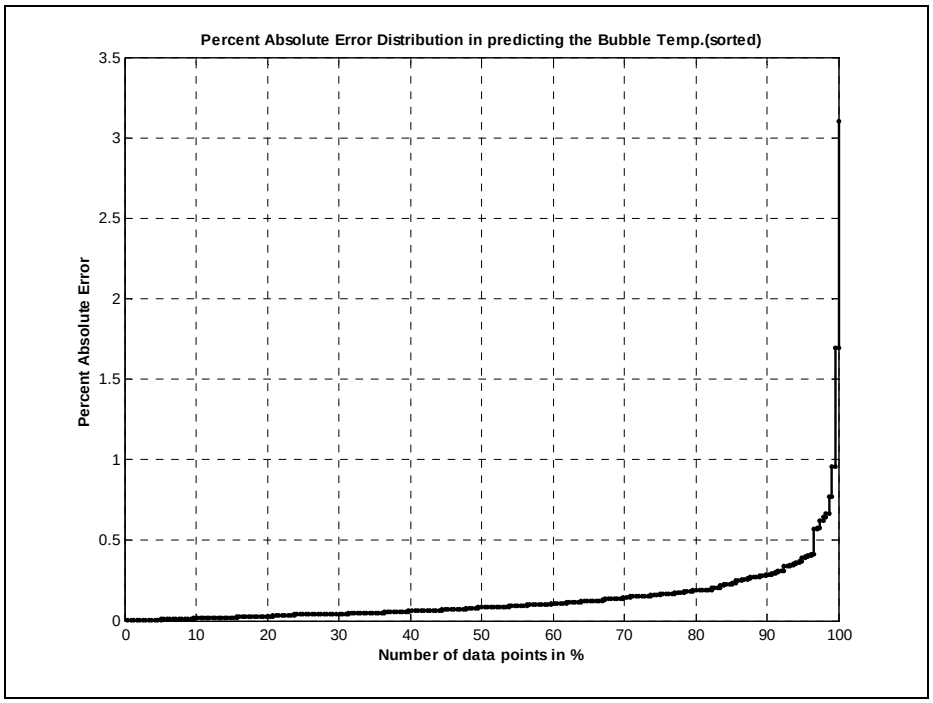

Figure 4-39 Percent absolute error distribution, *after feature extraction, predicting the bubble-point temperatures*

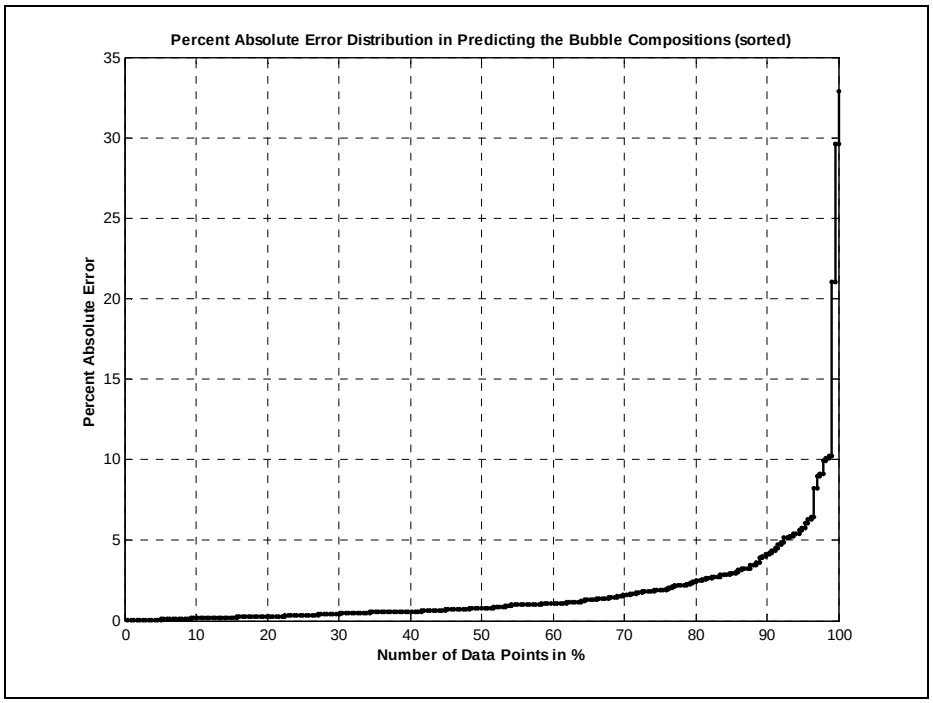

Figure 4-40 Percent absolute error distribution, *predicting the bubble compositions, after feature extraction*

The neural network model developed based on the extracted features gives regression R-value equal to 0.997 in the case of predicting the bubble temperatures and 0.998 in the case of predicting the bubble compositions. The bubble temperature predictions were with a percent absolute error less than 3.1%. More than 99% of predictions were made with percent absolute error less than unity. The average absolute error in predicting the bubble temperatures is equal to 0.4958 ^{o}C for temperatures ranging between 326.45 ^{o}C to 372.45 ^{o}C

The percent absolute error in predicting the bubble composition using the proposed model is shown in Figure 4-40. More than 99% of bubble composition predictions were with percent absolute error less than 10.3%. The average absolute error in predicting the bubble compositions is equal to 0.0103.

The neural network model obtained after applying the feature extraction method gives comparable but lower performance results as compared to the model base on all input features. Having simplified and reduced model which call for smaller number of pure component property data may be taken as an advantage. This result show that, attempt to reduce the input features using feature extraction methods is worth to consider whenever possible.

4.3 Neural network model for a drying process

Due to the numerous possible combinations of factors affecting the drying condition, the experimental measurements of drying rate data is time consuming. Mathematical models with predictive capability offer an alternative way of generating drying characteristic data within limitations. In the existing modeling techniques, estimation is based on all available data. The ability of these models to predict unseen data (data that was not used for curve fitting/ parameter estimation) has not been adequately established [22].

An alternative approach to modeling, and particularly to the description of drying kinetics, is the application of artificial neural networks. The problem of designing a mathematical model of a process using only observed data has attracted much attention. This thesis describes a predictive modeling approach based on artificial neural networks. A neural network model is appropriate for this kind of application due to its inherent capability to capture nonlinear relationships effectively.

4.3.1 Defining neural network model for the drying operation

Calculations concerning the prediction of moisture content of dried material in time for potato were made using MLP type neural networks. Inputs to the network were drying process parameters: air flow rate V_G and drying agent temperature at the inlet T_G to the apparatus, air relative humidity RH, initial moisture content X_{m0} as well as the total drying time i where we want to do the prediction. The output from the network is the moisture content of the material as related to the input values.

The experimental design is in such a way that range of inlet air flow rates and inlet air temperatures are covered. To train and verify the network, the whole set of experimental data was divided into three subsets: a training set, a testing set and verification set. So, the neural network model is designed to predict the moisture content of the material after a given time using the initial moisture content of the material, the inlet air temperature, the air flux and the air relative humidity as input. Figure 4-41 demonstrate the neural network model.

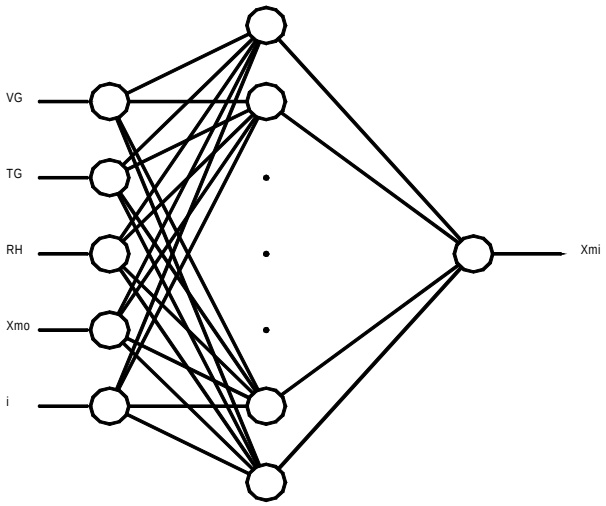

Figure 4-41 Graphical representation of the neural network model for the drying process

The experiments were carried out at variable process parameters V_G and T_G and RH. The performance of neural prediction was made by means of the regression R-value and the average absolute error.

4.3.2 Developing the neural network model

The computer program COMPARE.m designed for this particular problem will take the raw data, pre-process them, and form the required input-output format. Comparing different networks with different number of hidden neurons using this program gives the following results. From Figure 4-42 and Figure 4-43 we can see that 5-9-1 is the optimum neural network topology.

Further analyses on the performance of the 5-9-1 neural model are shown in Figure 4-44 and Figure 4-45. The regression R-value in using the proposed model is 0.999.From Figure 4-45 we can see that the maximum percent absolute error is equal to 12.3% and 95% of the total prediction were predicted with percent absolute error less than 6. The average absolute error in predicting the moisture content equal to 0.0059 % (wet basis) for moisture content ranging from 0.0906% to 0.8316%.

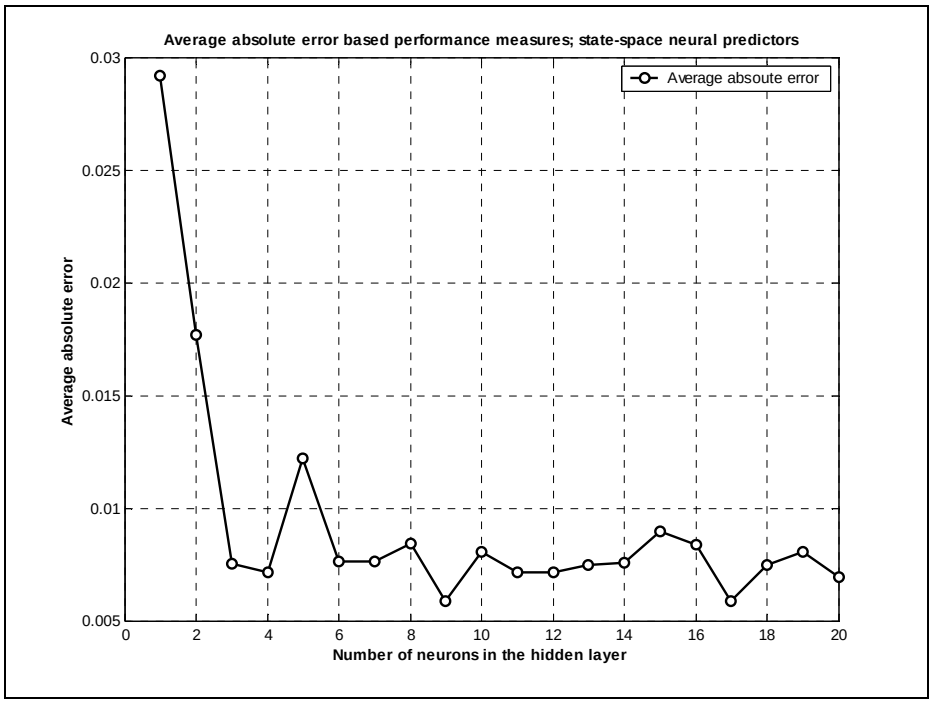

Figure 4-42 Average absolute error based performance measures, *Drying process*

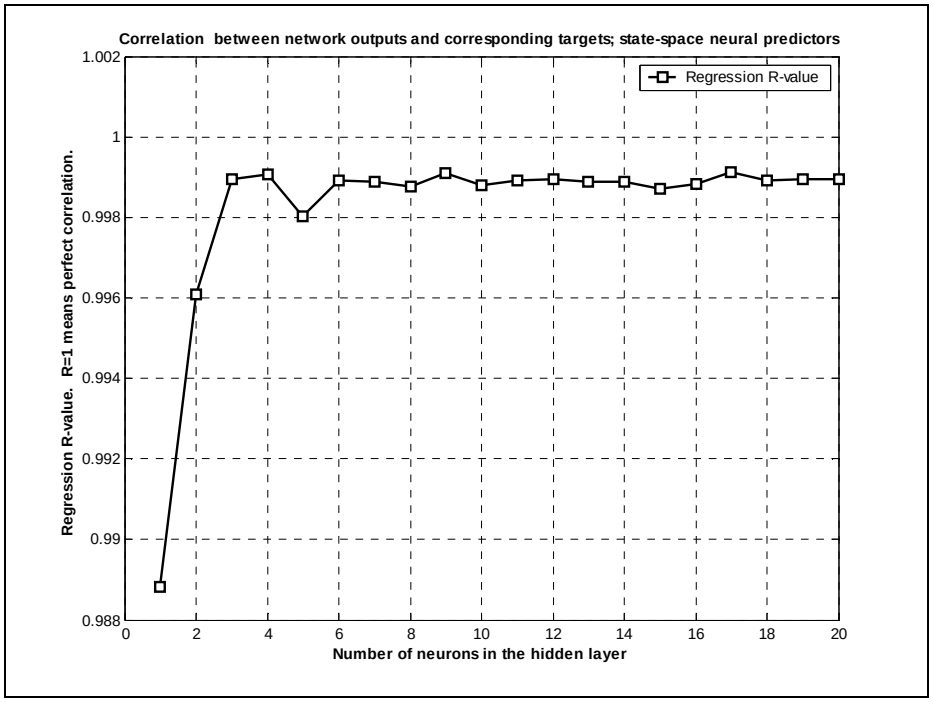

Figure 4-43 Regression R-value based performance measures, *Drying process*

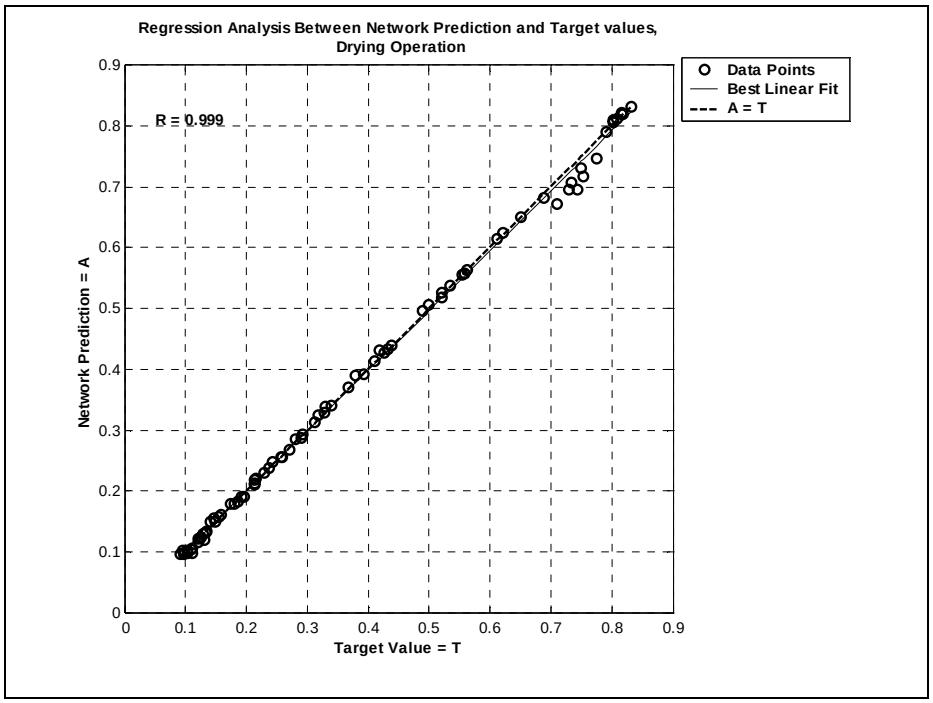

Figure 4-44 Regression analysis between neural prediction and target values, *using the proposed network topology, Drying process*

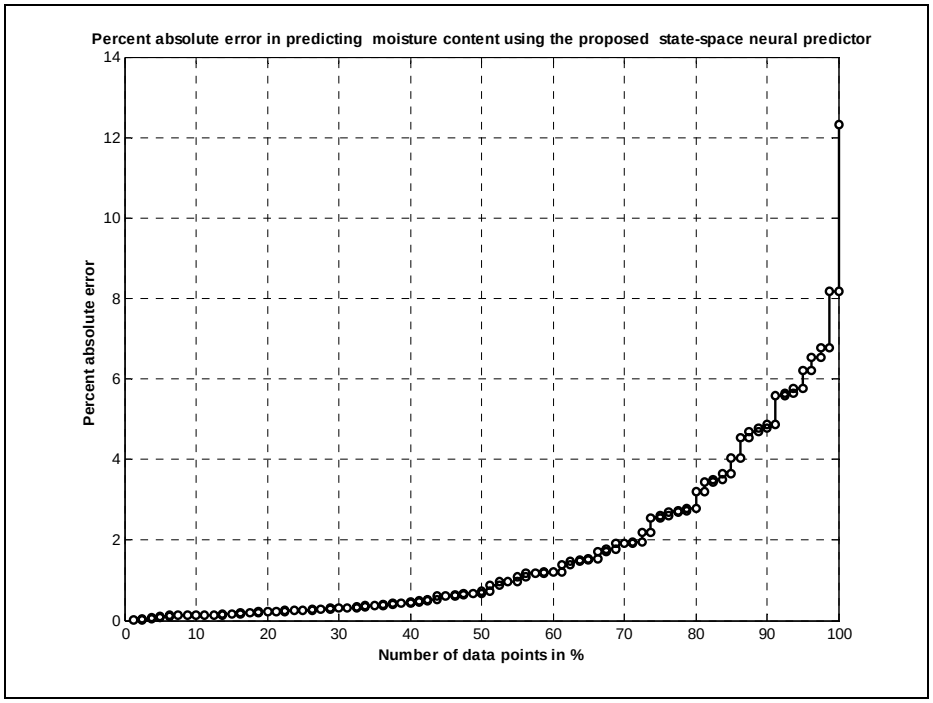

Figure 4-45 Percent absolute error distribution in predicting moisture content, *using the proposed network topology, Drying process*

4.3.3 Using the neural network model to analyze the drying characteristics of potato

The neural network prediction outputs are compared with the experimental values as shown in Figure 4-46 and Figure 4-47. There are totally eight separate experiments. As can be seen from the two figures the neural network predictions are excellent for each case.

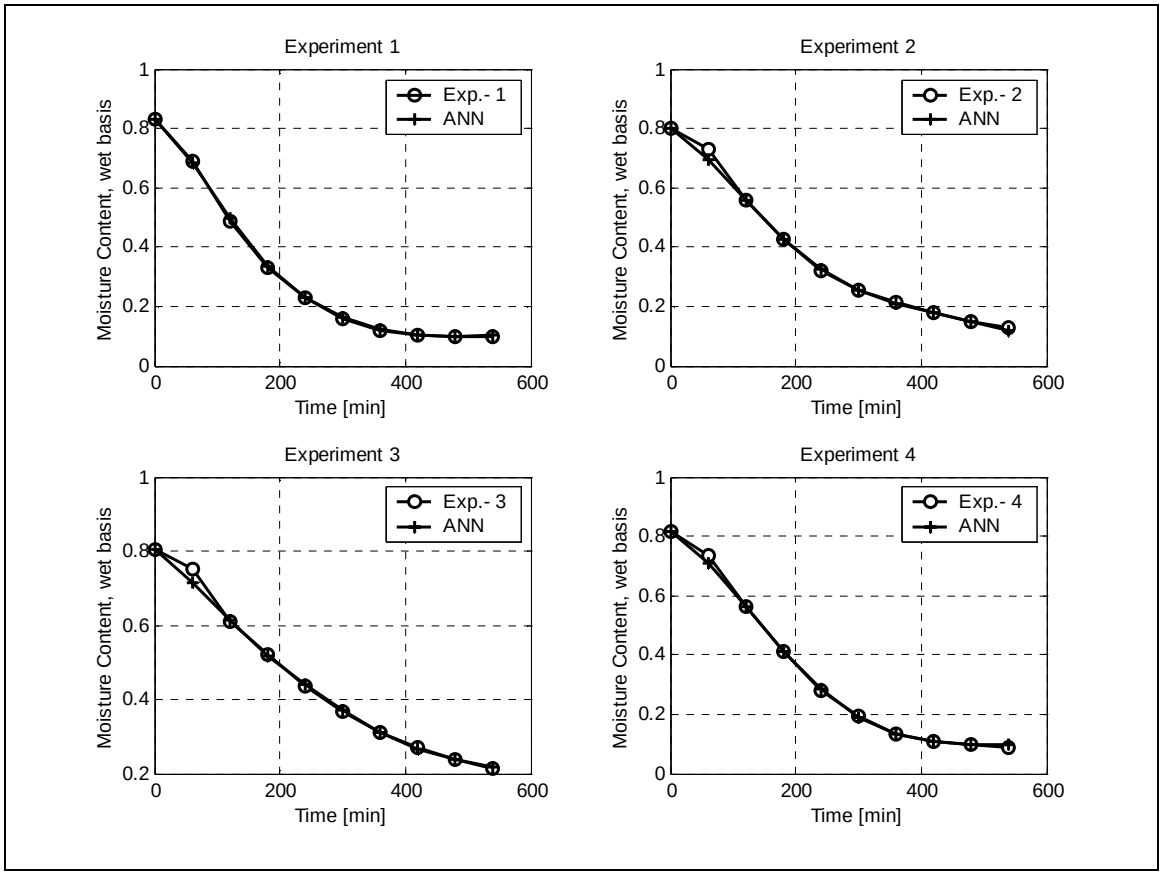

Figure 4-46 Comparison between the network prediction and the experimental values, *Potato drying*

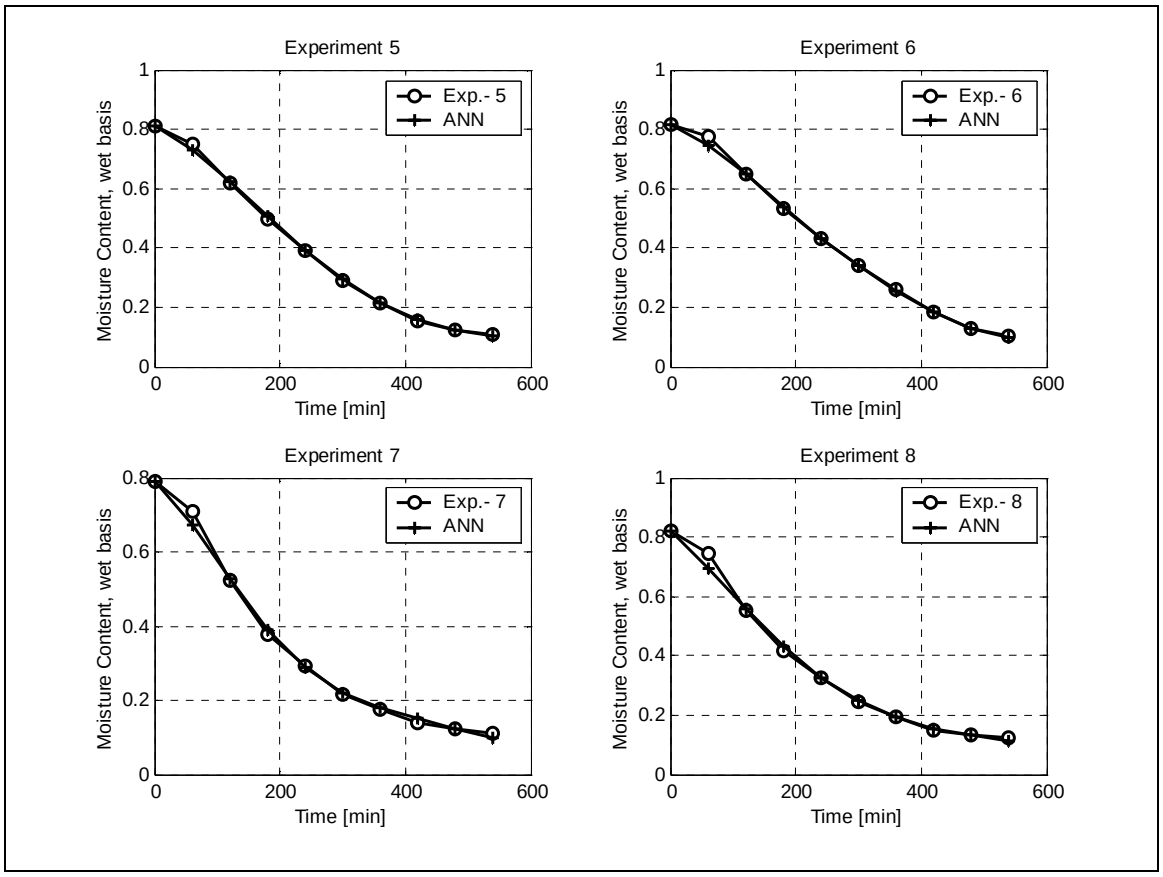

Figure 4-47 Comparison between the network prediction and the experimental values, *Continued*

The application of the neural network model to investigate the drying characteristics of the material is also investigated by taking input parameters that were not in the training data set. Such analysis is used to check the generality of the neural network model. Figure 4-48 shows the result obtained in using the neural network model to investigate the effect of incoming air temperature and Figure 4-49 shows the result obtained in investigating the effect of incoming air relative humidity.

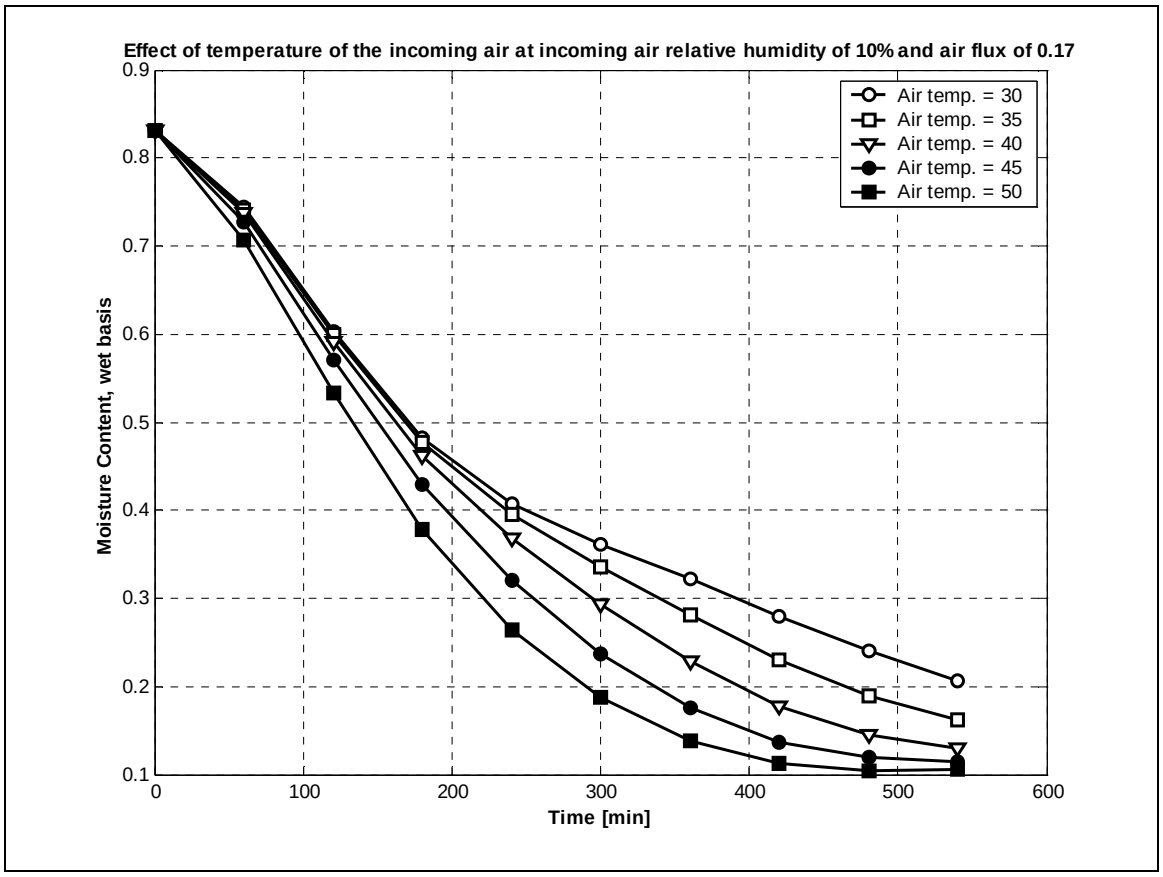

Figure 4-48 Effect of incoming air temperature. ***Simulated using the 5-9-1 neural network topology***

The neural network predictor gives a physically sound result on the effect of entering air temperature. Higher entering air temperature means higher drying rate and this is clearly shown on the above figure.

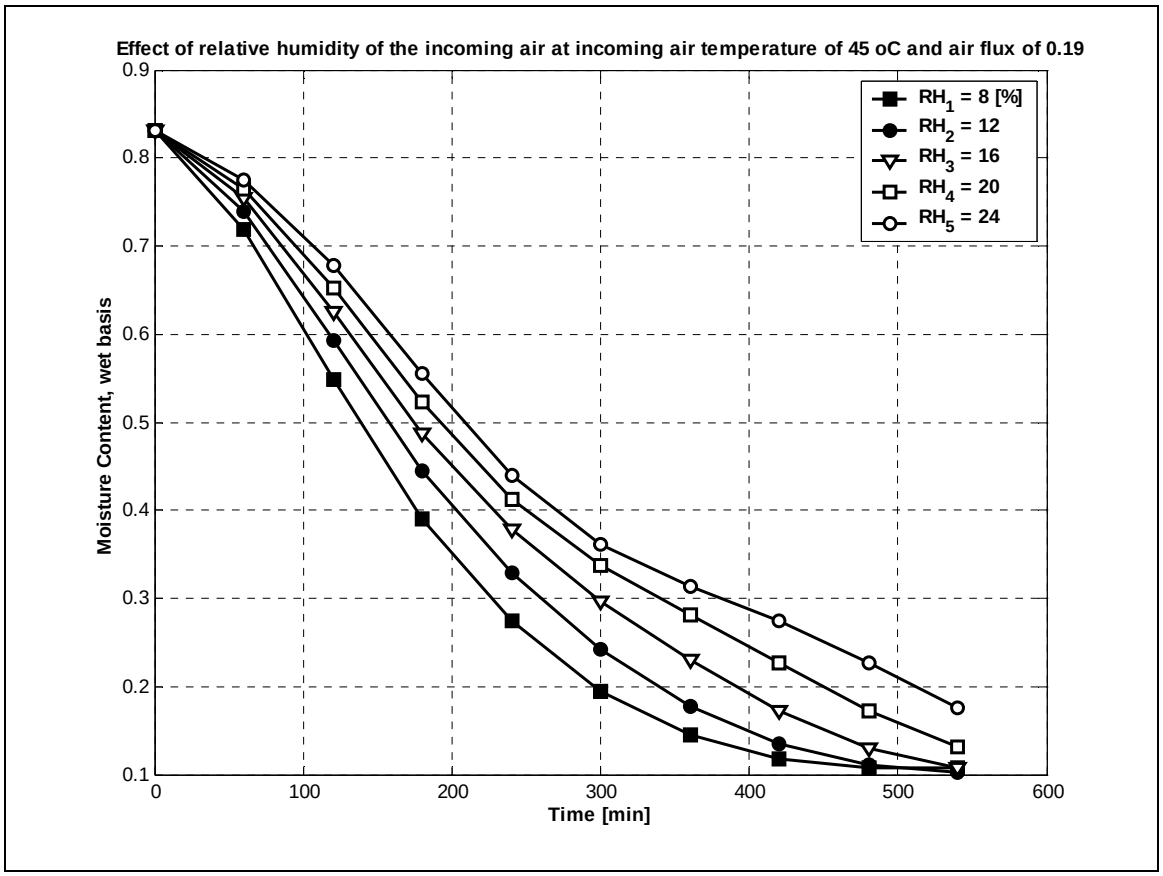

Figure 4-49 Effect of incoming air relative humidity. ***Simulated using the 5-9-1 neural network topology***

Higher relative humidity means lower drying rate and the neural network model clearly indicate this physical phenomenon as shown in Figure 4-49. Taking into account presented results; it was found that ability of the MLP network to predict changes in moisture content X_m of the material over time was very good.

4.4 Modeling the temperature-time profiles of adiabatic reaction

To obtain maximum yield from a reaction, a mode of heat transfer may have to be devised so that optimum temperature programming is realized. Since sensitivity of a reaction to temperature is one of major factors in the selection of the reactor type as well as the heat transfer mode, it would be desirable to quantify this characteristic [22]. Processing, design, quality control, and operational applications of systems using energetic materials, e.g. explosives and propellants, all require an understanding of thermal hazards and an ability to predict safety limits [23]

Calculation of adiabatic temperature-time curve of the reaction progress can be used to determine the decrease of the thermal stability of materials during storage at temperatures at which the reaction will begin. Due to insufficient thermal convection and limited thermal conductivity, a progressive temperature increase in the sample can easily take place, resulting in a thermal explosion [23].

Numerous methods have been presented for prediction of the reaction progress of exothermic reactions under adiabatic condition [22]. However, because industrially important reactions usually have a multi-step nature, the accurate determination of the kinetic characteristics strongly influences the ability to correctly describe the progress of the reaction. For adiabatic reactions, incorrect kinetic description of the process is usually the main source of series errors in its interpretation [23]. In this thesis work an attempt is made to use artificial neural networks to predict the temperature-time curves of an adiabatic reaction. A feedforward type neural network model was developed based on temperature-time data obtained for adiabatic reactions with three different starting temperatures [24]. The three temperature-time profiles are thought to be used for the network training and the model is supposed to give the temperature-time profiles for other starting temperatures that were not in the training set.

4.4.1 Developing the neural network model

The first step is the data integration step. In this step the experimentally obtained data sets for the three cases were separately used and for each of them separate neural networks were developed. These networks were used to generate data at a known pre-specified time step. Then the generated data were collected together to give a single data set for network training. The temperature-time curves for the three starting temperatures are shown below.

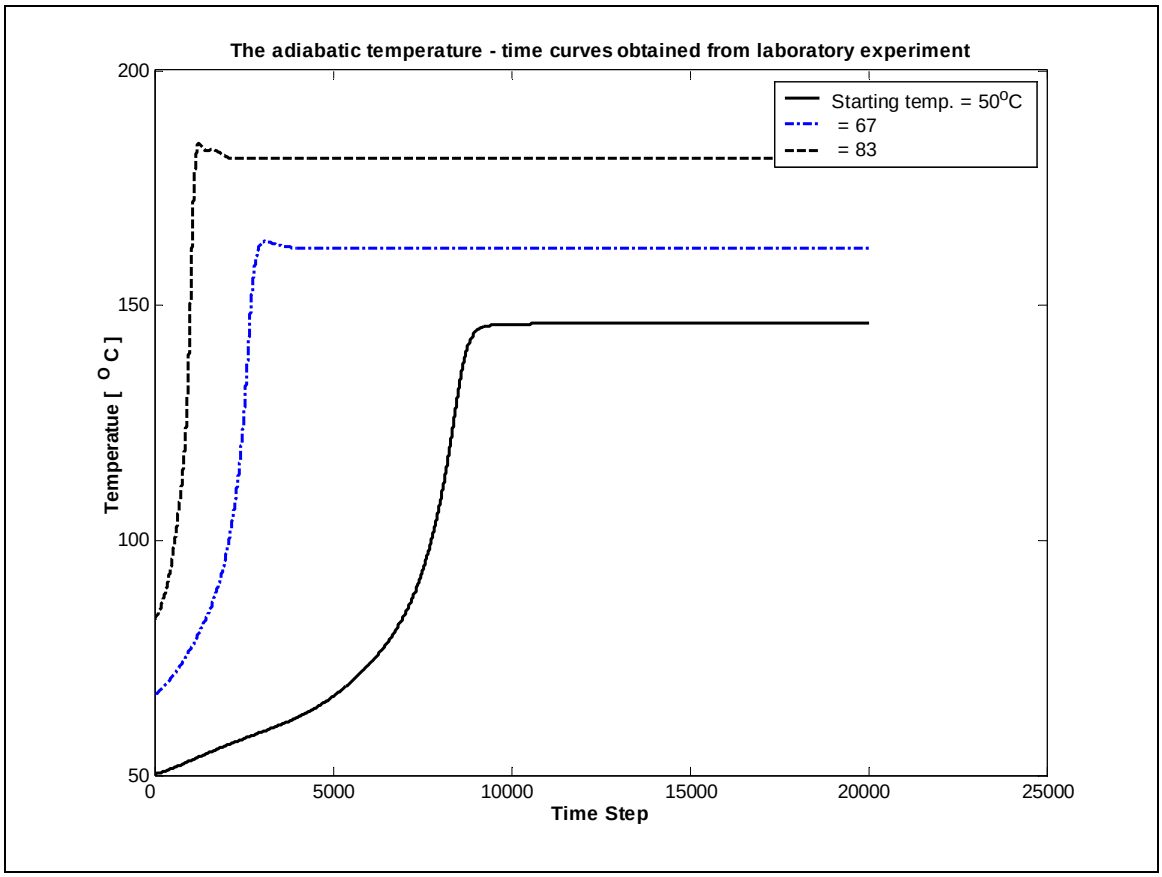

Figure 4-50 Adiabatic temperature-time curves for three different starting temperatures,

The computer program COMPARE.m, take the raw data pre-process them and form the required input-output format. Comparing different networks with different number of hidden neurons using this program gives the following results. The model development is in such a way that, it will take time in seconds after the start of the adiabatic reaction and starting temperature as input to the network and the temperature of the reaction mixture

as output. Comparing group of competing neural network topologies under different network parameters was done. Using sigmoid type activation function in both layers is better in performance.

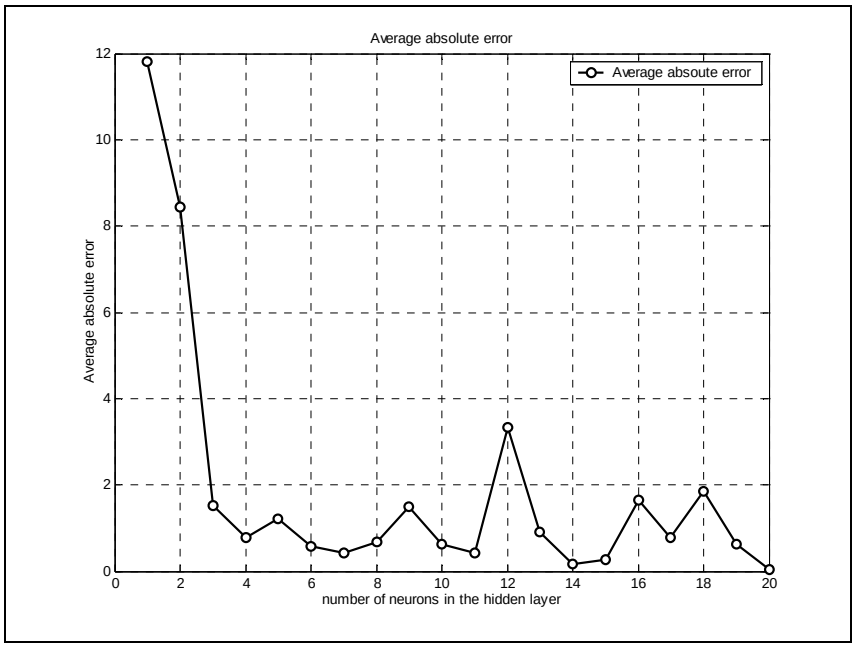

Figure 4-51 Average absolute error based performance measures, *Adiabatic reaction*

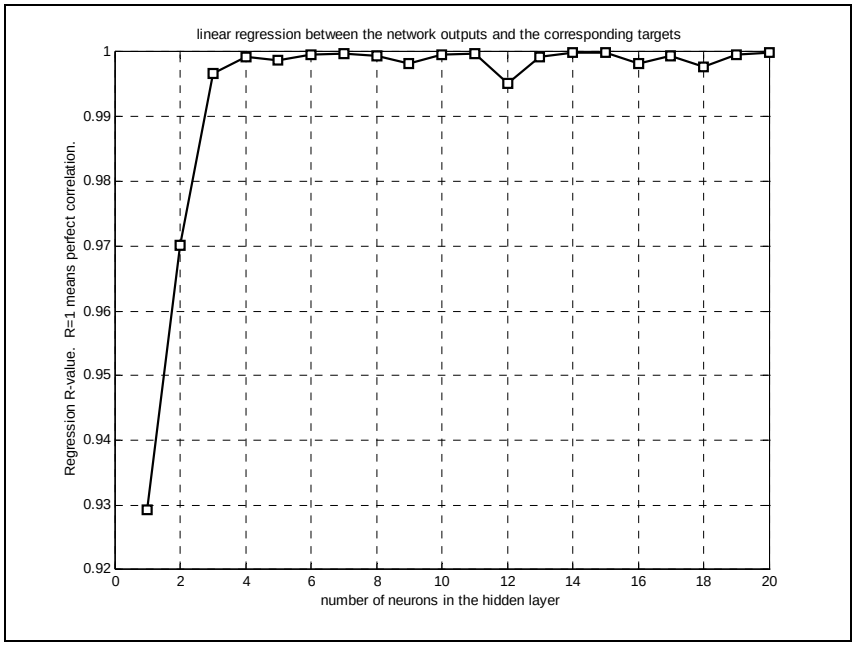

Figure 4-52 Regression R-value based performance measures, *Adiabatic reaction*

From Figure 4-51 and Figure 4-52 we can see that 2-7-1 topology as the optimum. Further analyses on the performance of the 2-7-1 neural model are shown in Figure 4-53 and Figure 4-54. The regression R-value in using the proposed model is almost unity.

From Figure 4-54 we can see that the maximum percent absolute error is equal to 2.8% and 98% of the total prediction were predicted with percent absolute error less than 1.8. The average absolute error in predicting the adiabatic temperature is equal to 0.32 ^{o}C for temperatures ranging from 50.14 ^{o}C to 184.58 ^{o}C .

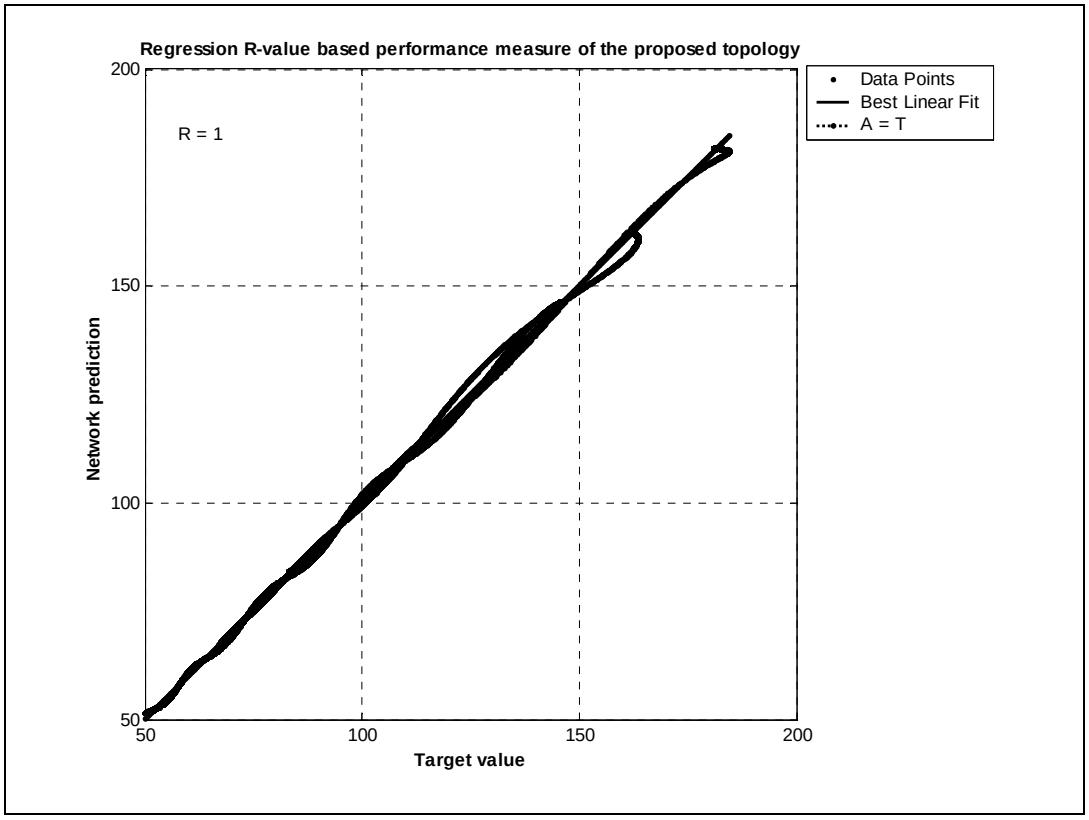

Figure 4-53 Regression analysis between neural prediction and target values, *using the proposed network topology, Adiabatic reaction*

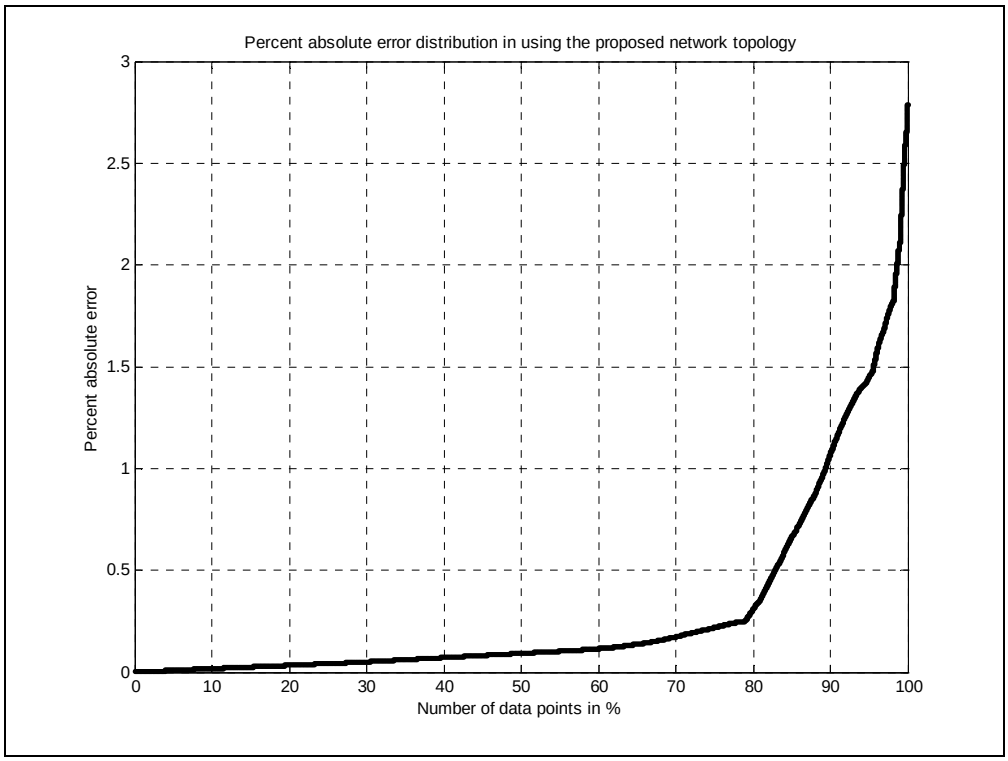

Figure 4-54 Percent absolute error distribution in predicting temperatures, *using the proposed network topology, Adiabatic reaction*

4.4.2 Using the neural network model to predict the temperature-time profiles of adiabatic reactions at different starting temperatures

The applications of the neural network model to investigate the temperature-time curves of adiabatic reactions for different starting temperatures were also considered, and analyzed based on its physical implication.

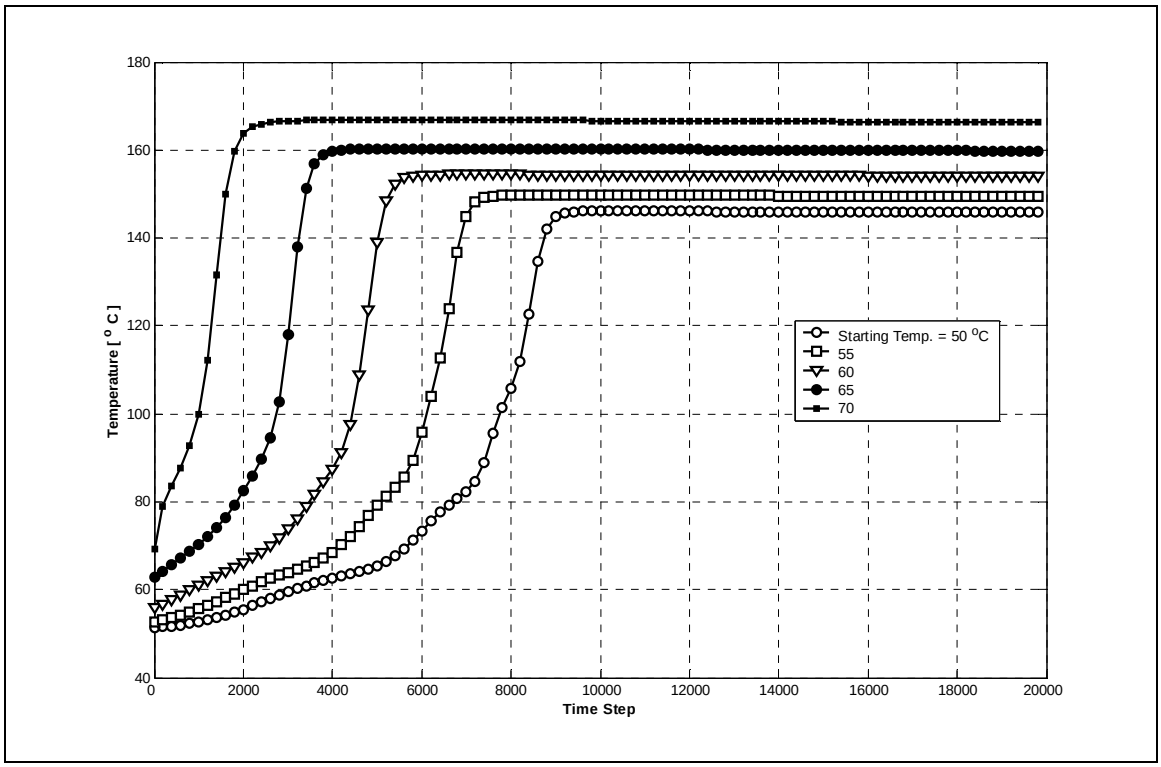

Figure 4-55 The adiabatic temperature-time curves for different starting temperatures. *Simulated using the 2-7-1 neural network topology*

The neural network predictor gives a consistent result for the above cases. The expected trend of the temperature-time curves are physically sound as can be seen from the above figure. The early (induction), intermediate (acceleratory), and late (decay) periods of the adiabatic reactions are well followed. Higher starting temperature means shorter induction period and this is clearly shown on the above figure.

Further analysis on using the proposed neural network model to predict the temperature-time profiles at starting temperature greater than 70 ^{o}C show some irregularity as can be seen in Figure 4-56.

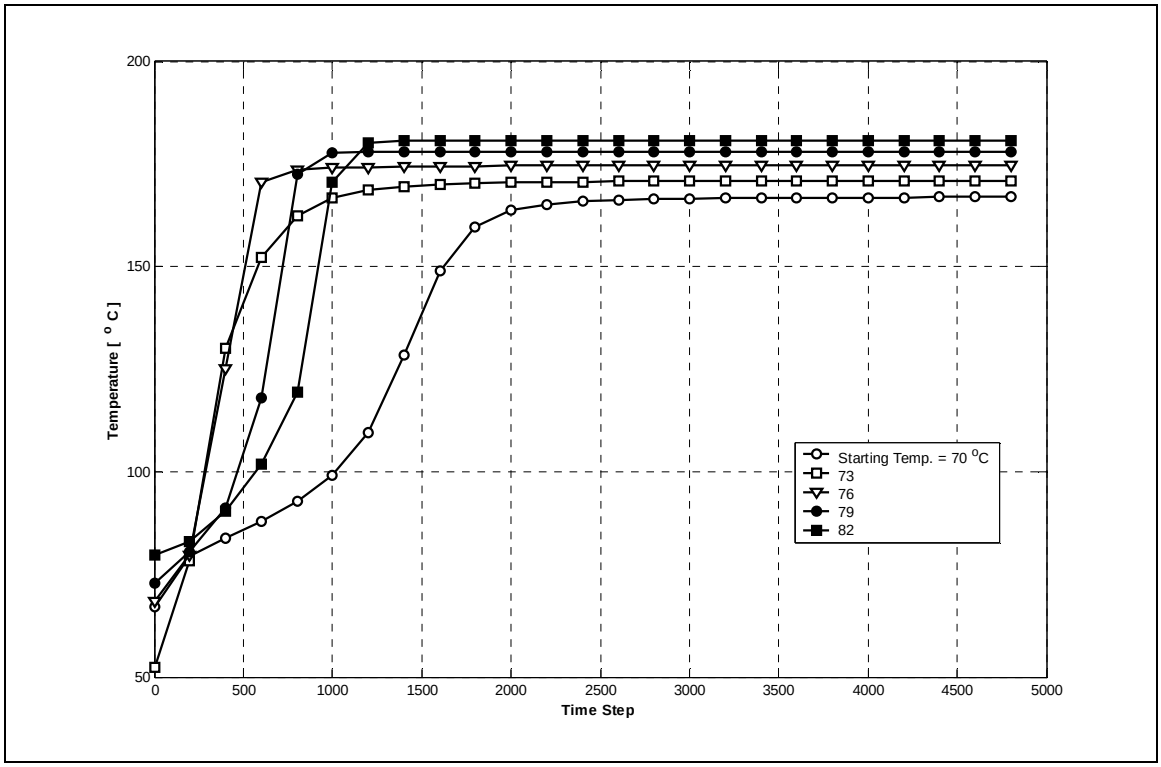

Figure 4-56 The adiabatic temperature-time curves for different starting temperatures. *Simulated using the 2-7-1 neural network topology*

The irregularity is due to the small number of cases used in the network training phase. The cases used were data obtained for three starting temperatures. The neural network developed based on these three cases looks inconsistent in predicting the temperature-time curves for starting temperatures greater than 70^{o}C.

Though the result obtained are a bit contradictory for this particular case, this thesis work confirm the potential of using neural network based models to predict the temperature-time curves for any starting temperatures.

4.5 Modeling an operator controlling a chemical plant

Here an artificial neural network is applied to simulate a human operator controlling a chemical plant. The data used to train a neural network is obtained from the *IEE Neural Networks Council Standards Committee, Working group on Data Modeling Benchmarks* [11]. The work in this thesis includes data pre-processing, feature extraction and network construction steps.

4.5.1 Problem description

Suppose you are given data from how a human operator controls a chemical plant. In particular, suppose that the operator has indications of monomer concentration (u1), change in monomer concentration (u2), monomer flow rate (u3), some local temperature changes in the plant (u4, u5), and with these makes decisions on how to set the set point for the monomer flow rate (y). Here, the actual value of the monomer flow rate to be put into the plant is controlled by a PID controller. Below, table 4-2 show portion of the data used for the network training.

Table 4-2 Sample data showing how an operator control a chemical plant

U1	U2	U3	U4	U5	Y
6.800	-0.050	401.000	-0.200	-0.100	500.000
6.590	-0.210	464.000	-0.100	0.100	700.000
6.590	0.000	703.000	-0.100	0.100	900.000
6.500	-0.090	797.000	0.100	0.100	700.000
6.480	-0.020	717.000	-0.100	0.100	700.000
6.540	0.060	706.000	-0.200	0.100	800.000
6.450	-0.090	784.000	0.000	0.100	800.000
6.450	0.000	794.000	-0.200	0.100	800.000
6.200	-0.250	792.000	0.000	0.000	1000.000
6.020	-0.180	1211.000	0.000	0.100	1400.000
5.800	-0.220	1557.000	-0.200	0.000	1600.000

The goal here is to construct a neural network model to simulate the operator. The data pre-processing steps include data transformation using the linear transformation method. Neural networks with different parameters were compared for their performance and the best one is chosen.

4.5.2 Operator simulation using all feature vectors

This section aimed to construct a neural network which uses all features as input to the network. So, the network in this case will have five input nodes. The data pre-processing step is linear.

In the network construction steps choosing a good training algorithm, activation functions, and the number of hidden neurons to be used are the main activities. The Levenberg-Marquardt backpropagation gives the best performance when compared to Gradient descent backpropagation, Gradient descent with momentum backpropagation, Gradient descent with adaptive learning rate backpropagation, and Gradient descent with momentum & adaptive learning rate backpropagation. As to the activation functions, using logsig in the hidden layer and linear in the output layer gives the best performance results.

Comparing networks with different number of hidden neurons is shown in figure bellow. And the selection is in such a way that the performance together with simplicity of the model is considered together as done in the previous problems.

The network with a topology of 5-7-1 appears to be the optimum network, as can be seen from Figure 4-57 and Figure 4-58. Further analysis on the performance of the proposed network topology was done and results are shown in Figure 4-59 and Figure 4-60.

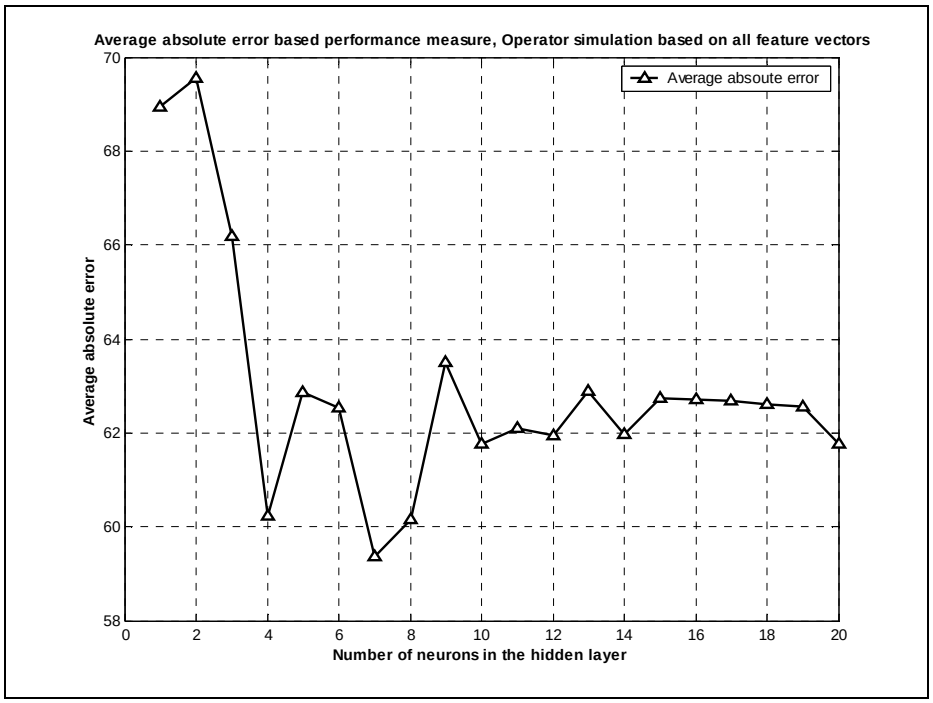

Figure 4-57 Average absolute error based performance. *Operator simulation based on all feature vectors*

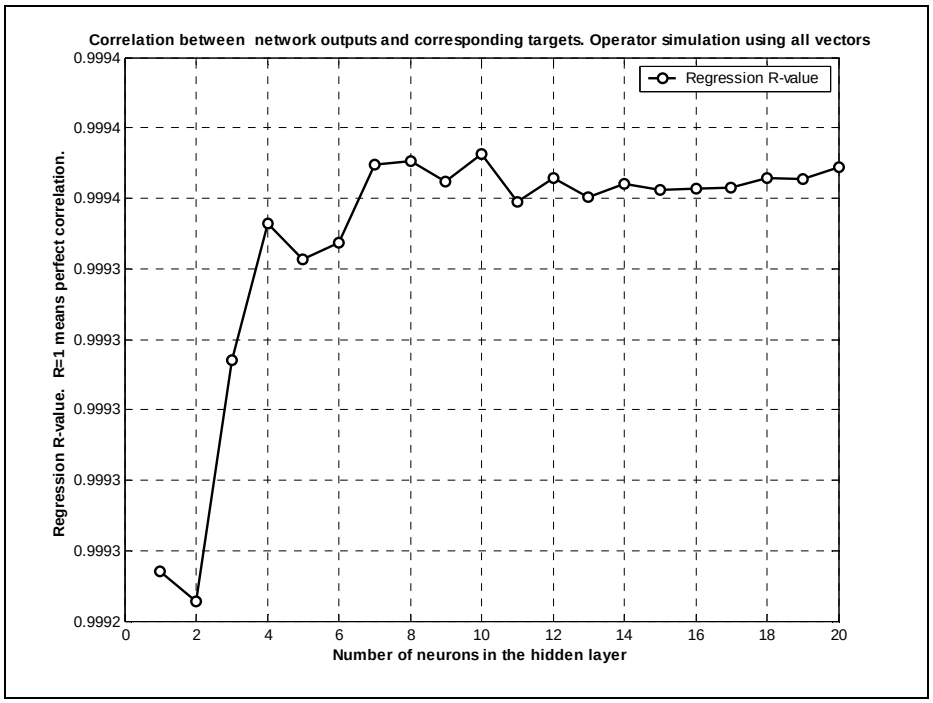

Figure 4-58 Regression R-value based performance measures. *Operator simulation using all feature vectors*

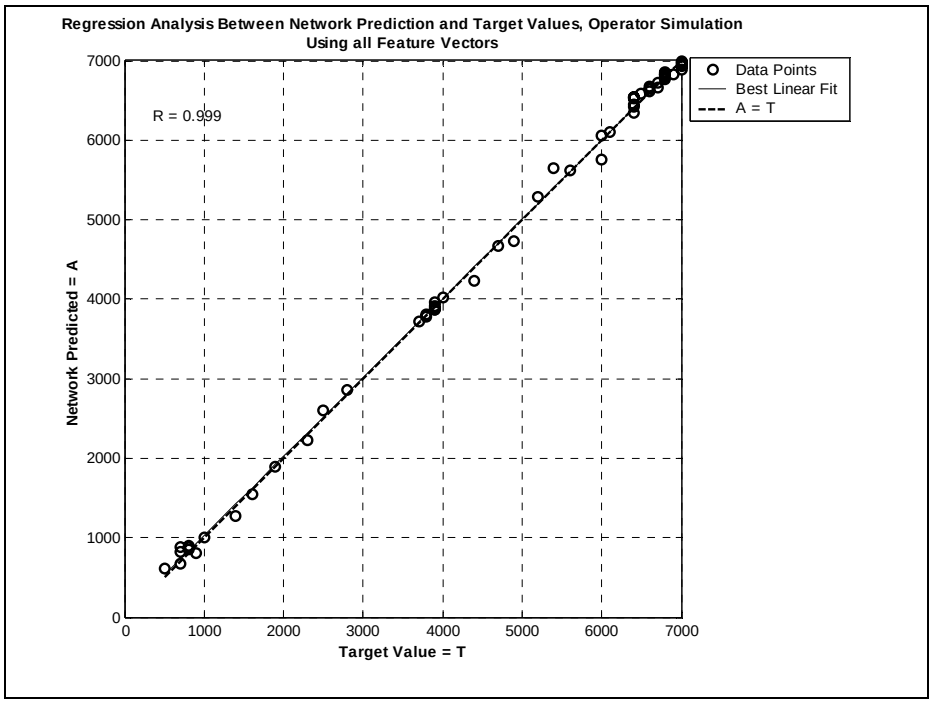

Figure 4-59 Correlation between the neural prediction and the target values, Operator simulation using all feature vectors

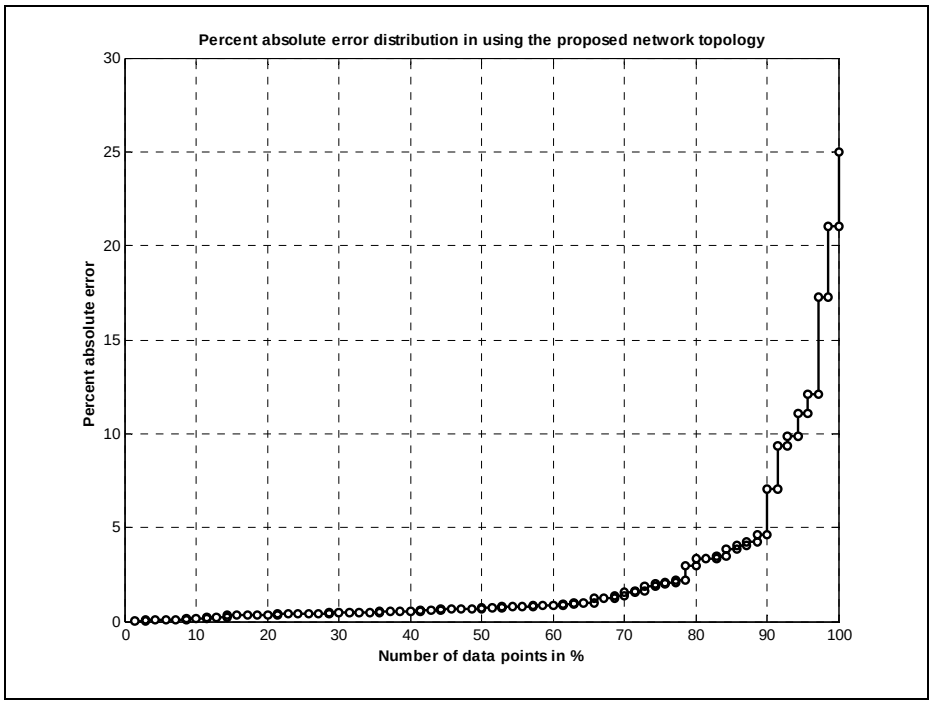

Figure 4-60 Percent absolute error distribution in using the proposed network topology, *operator simulation using all feature vectors*

The proposed network predictor performs under percent absolute error less than 11% for more than 95% of the prediction. Only five data points are predicted with a percent absolute error greater than 15. The regression R-value is equal to 0.9994. The average absolute error in using the proposed model to predict the set pint is equal to 59.3581 for set point values ranging from 500 to 7000.

Figure 4-52 below shows how the proposed model and the human operator track the set point changes required by the PID controller for the proper functioning of the chemical plant. The neural network based prediction model gives almost the same function as the experienced human operator use his experiential knowledge to decide on the value of a set point based on collected input parameters.

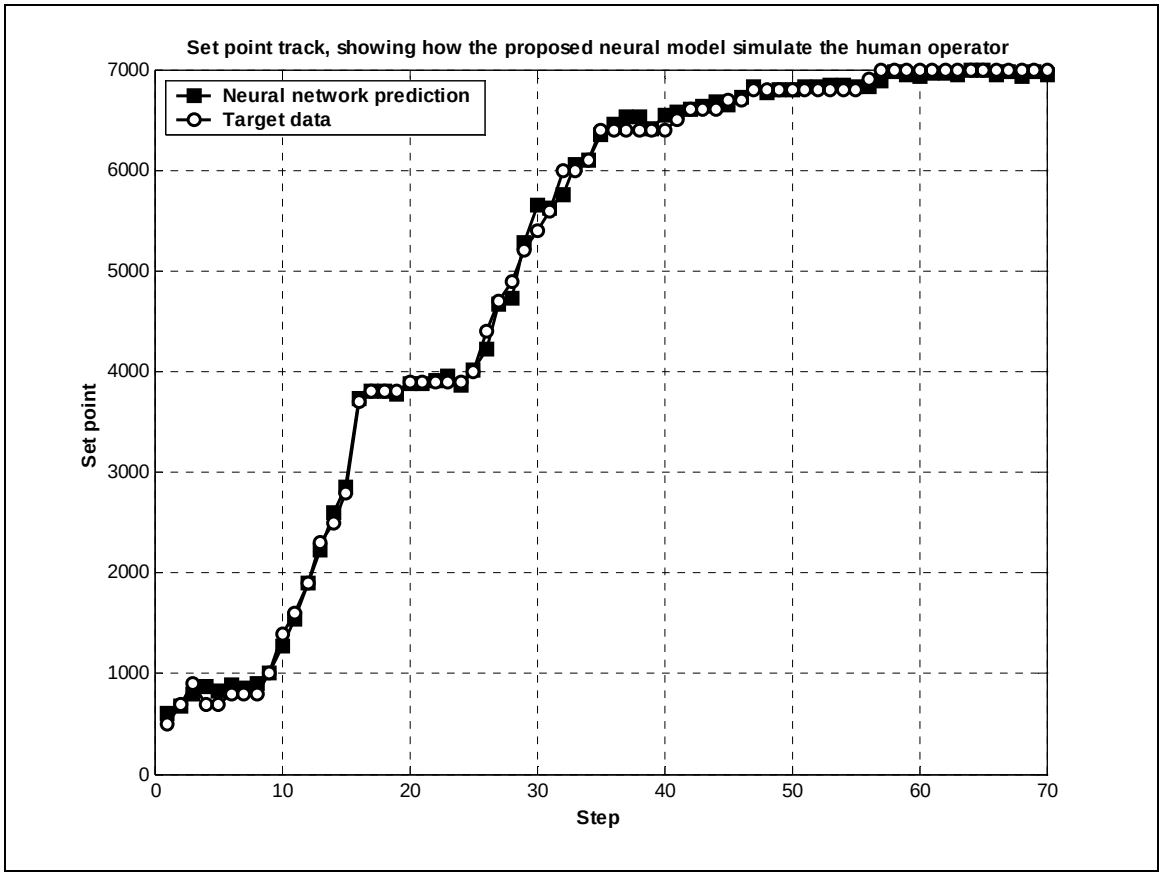

Figure 4-61 Set point track, *showing how the proposed neural model simulate the human operator*

4.5.3 Operator simulation after feature extraction

The Belue-Bauer method of selecting features is applied on the operator data to extract the most relevant features in controlling the chemical plant. The result is shown in figure below.

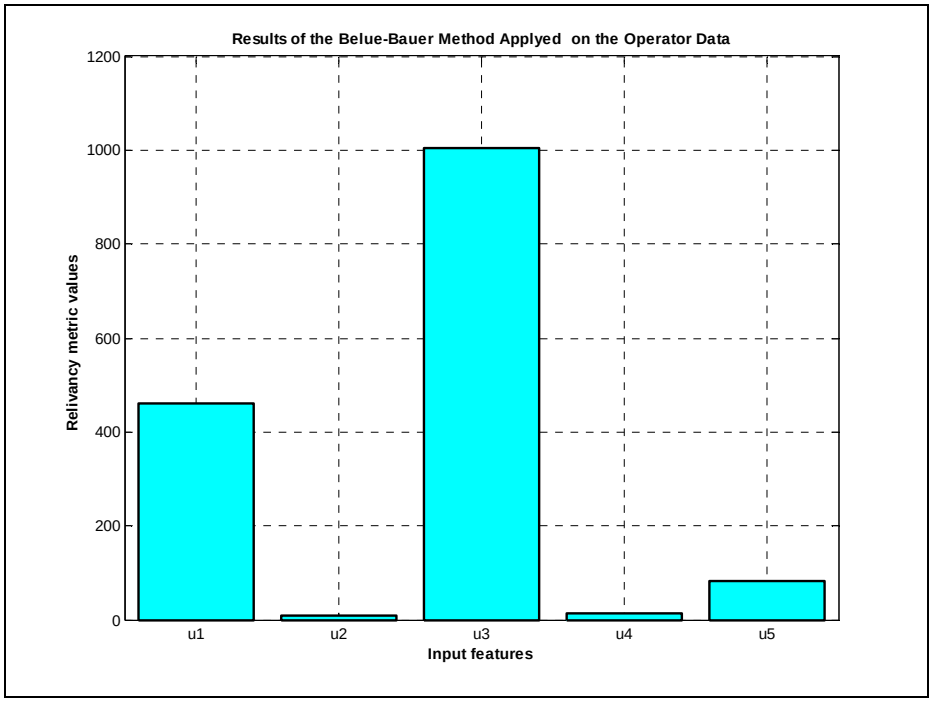

Figure 4-62 Results of the Belue-Bauer feature extraction method for the operator data

Figure 4-62 shows that features 1 and 3 are more relevant than the other. Taking these two features as input to network modeling will show us if there is any advantage in applying the feature extraction method for this problem. Comparing a set of competing neural network models is done as in previous sections. Using sigmoid activation function in the hidden layer and linear type activation function in the output layer gives a good performance over that of using sigmoid activation function in both layers.

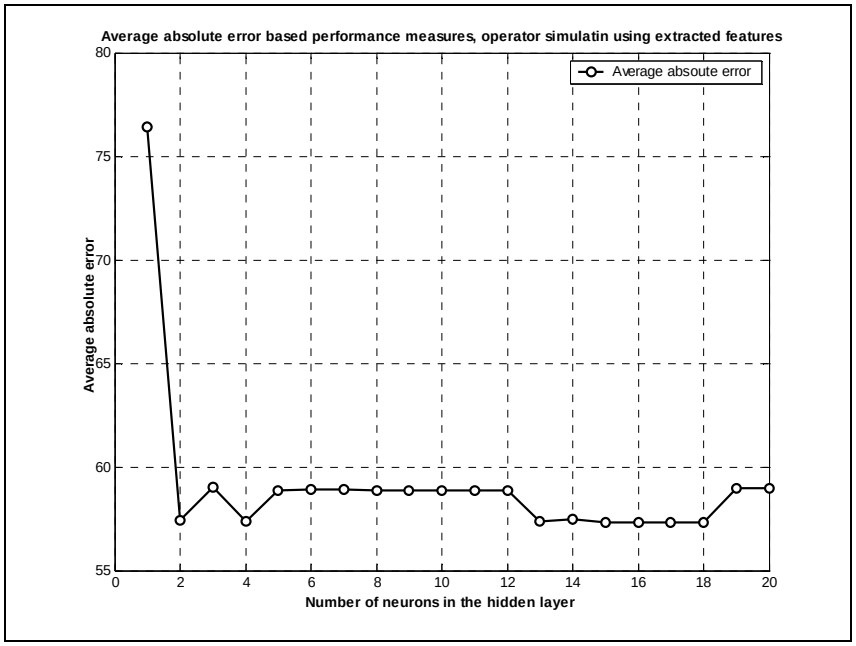

Figure 4-63 Average absolute error based performance measures, *Operator simulation using extracted features- {u1, u3}*

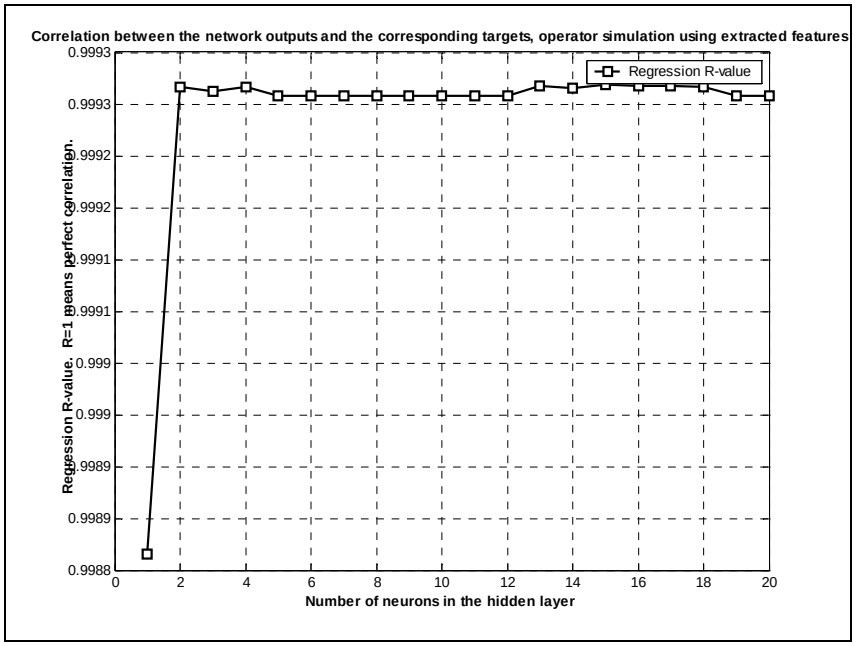

Figure 4-64 Regression R-value based performance, *Operator simulation using extracted features- {u1, u3}*

As can be seen from the above results using the feature extraction method reduce the average absolute error obtainable. In this case a relatively simple network topology which

needs fewer imputes than the previous model is obtained by applying the feature extraction method. The optimum topology is 2-2-1 and the average absolute error in predicting the set point for the controller using this topology is equal to 57.44 for set point values ranging from 500 to 7000. Figure 4-65 show that the regression R-value in using the proposed model to predict the set point equal to 0.999.

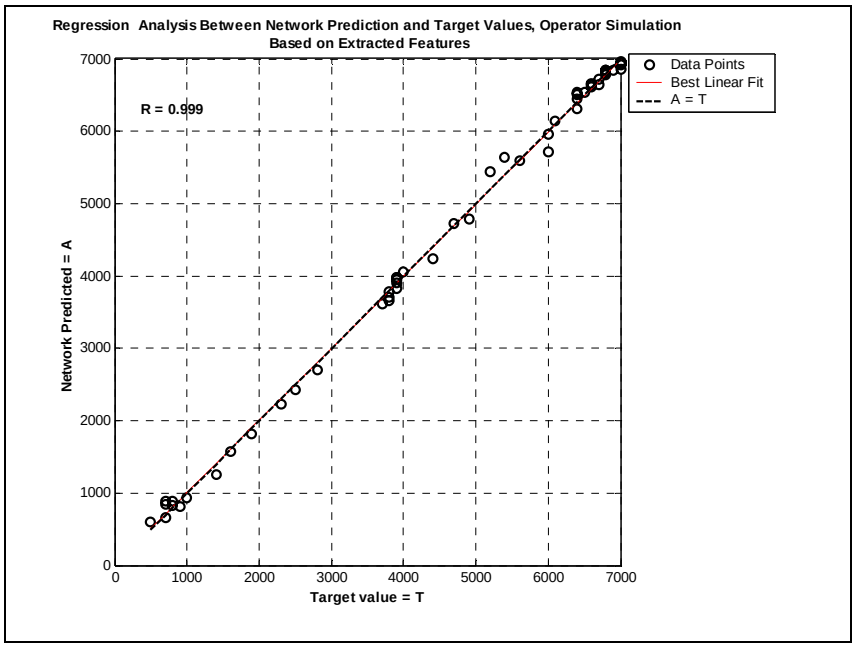

Figure 4-65 Correlation analysis for the proposed network topology, *Operator simulation, based on extracted features*

The percent absolute error distribution shown in Figure 4-66 indicate that 95% of the predictions were within percent absolute error of less than 10.6%. The prediction performance is demonstrated by comparing the network prediction and the data obtained from the human operator as shown in Figure 4-67.

The neural network model obtained after applying the feature extraction method gives better performance results as compared to the model base on all input features. Having simplified and reduced model which call for smaller number of measurement parameters as input is a good improvement. Using measurements of monomer concentration (*u1*) and monomer flow rate (*u3*) as input is enough for this simplified model.

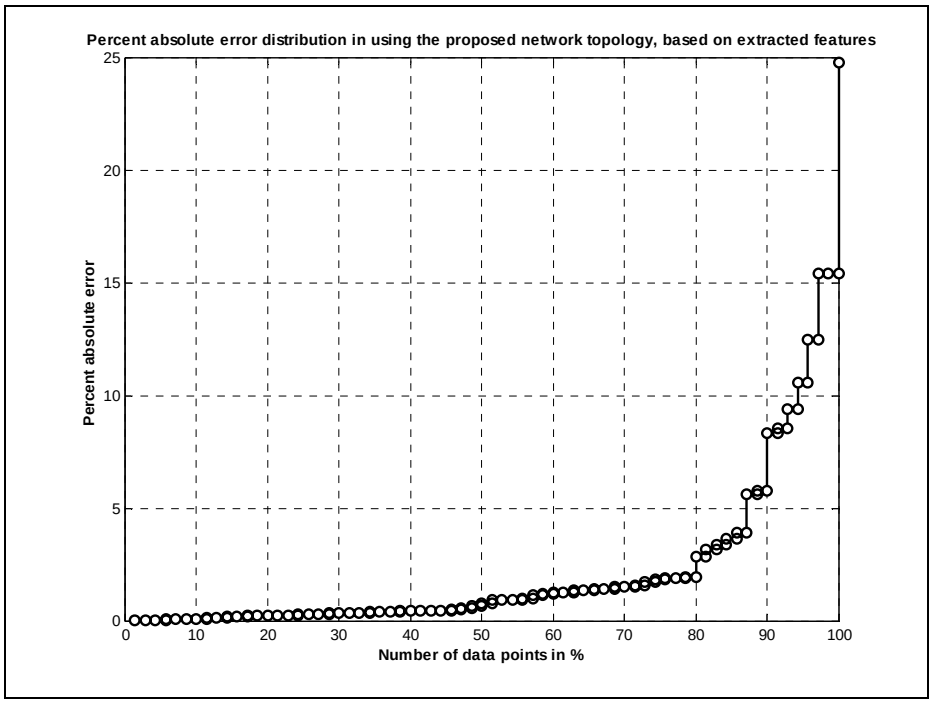

Figure 4-66 Percent error distribution in using the proposed neural network, *based on extracted features*

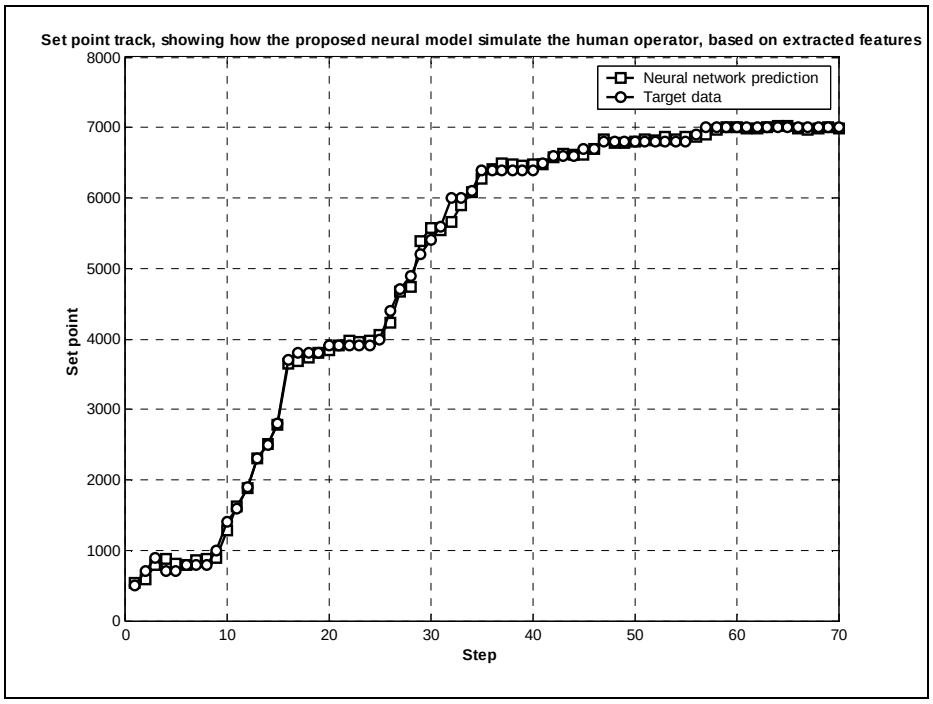

Figure 4-67 Set point track, showing how the proposed model simulate the human operator, *based on extracted features*

4.6 Neural network based CO_2 effluent analyzer

In time series problems, the objective is to predict ahead the value of a variable that varies in time; using previous values of that and/or other variables (see Bishop, 1995). Typically the predicted variable is continuous, so that time series prediction is usually a specialized form of regression [18]

4.6.1 Problem description

It is also usual to predict the next value in a series from a fixed number of previous values (looking ahead a single time step). When the next value in a series is generated, further values can be estimated by feeding the newly-estimated value back into the network together with other previous values: time series projection. Obviously, the reliability of projection drops the more steps ahead one tries to predict, and if a particular distance ahead is required, it is probably better to train a network specifically for that degree of look ahead.

Any type of network can be used for time series prediction (the network type must, however, be appropriate for regression or classification, depending on the problem type). The network can also have any number of input and output variables. However, most commonly there is a single variable that is both the input and (with the lookahead taken into account) the output. Configuring a network for time series usage alters the way that data is pre-processed (i.e., it is drawn from a number of sequential cases, rather than a single case), but the network is executed and trained just as for any other problem.

The problem sited in this thesis is based on furnace data file obtained from the *IEE Neural Networks Council Standards Committee, Working group on Data Modeling Benchmarks* [11]. Sample of this data is shown in table 4-3 below. The goal of this thesis is to use this data to train a neural network so that the trained network can be used in time series forecasting requirements. There are originally 296 data points {y(t),u(t)}, from t=1 to t=296. y(t) is the output CO_2 concentration and u(t) is the input gas flow rate. Here we are trying to predict y(t) based on {y(t-1), y(t-2), y(t-3), y(t-4), u(t-1), u(t-2), u(t-3), u(t-

4), u(t-5), u(t-6)}. This reduces the number of effective data points to 290. The first column below is the output variable y(t), the remaining columns are the input variables {y(t-1), y(t-2),...u(t-6)}. output y(t); input y(t-1) y(t-2) y(t-3) y(t-4) u(t-1) u(t-2) u(t-3) u(t-4) u(t-5) u(t-6) ;

Table 4-3 Portion of the data used to train the CO_2 concentration predictor

Y(T)	Y(T-1)	Y(T-2)	Y(T-3)	Y(T-4)	U(T-1)	U(T-2)	U(T-3)	U(T-4)	U(T-5)	U(T-6)
52.70	53.10	53.40	53.50	53.50	0.44	0.37	0.34	0.18	0.00	-0.11
52.40	52.70	53.10	53.40	53.50	0.46	0.44	0.37	0.34	0.18	0.00
52.20	52.40	52.70	53.10	53.40	0.35	0.46	0.44	0.37	0.34	0.18
52.00	52.20	52.40	52.70	53.10	0.13	0.35	0.46	0.44	0.37	0.34
52.00	52.00	52.20	52.40	52.70	-0.18	0.13	0.35	0.46	0.44	0.37
52.40	52.00	52.00	52.20	52.40	-0.59	-0.18	0.13	0.35	0.46	0.44
53.00	52.40	52.00	52.00	52.20	-1.06	-0.59	-0.18	0.13	0.35	0.46
54.00	53.00	52.40	52.00	52.00	-1.42	-1.06	-0.59	-0.18	0.13	0.35
54.90	54.00	53.00	52.40	52.00	-1.52	-1.42	-1.06	-0.59	-0.18	0.13
56.00	54.90	54.00	53.00	52.40	-1.30	-1.52	-1.42	-1.06	-0.59	-0.18
56.80	56.00	54.90	54.00	53.00	-0.81	-1.30	-1.52	-1.42	-1.06	-0.59
56.80	56.80	56.00	54.90	54.00	-0.48	-0.81	-1.30	-1.52	-1.42	-1.06
56.40	56.80	56.80	56.00	54.90	-0.19	-0.48	-0.81	-1.30	-1.52	-1.42
55.70	56.40	56.80	56.80	56.00	0.09	-0.19	-0.48	-0.81	-1.30	-1.52
55.00	55.70	56.40	56.80	56.80	0.44	0.09	-0.19	-0.48	-0.81	-1.30
54.30	55.00	55.70	56.40	56.80	0.77	0.44	0.09	-0.19	-0.48	-0.81

4.6.2 Neural network model based on all features

The first stage in preparing the appropriate network model is to pre-process the data before using it in the training stage. In the data pre-processing step the data is first transformed using a linear transformation method. Comparing different neural network models based on different parameters are shown in figures bellow

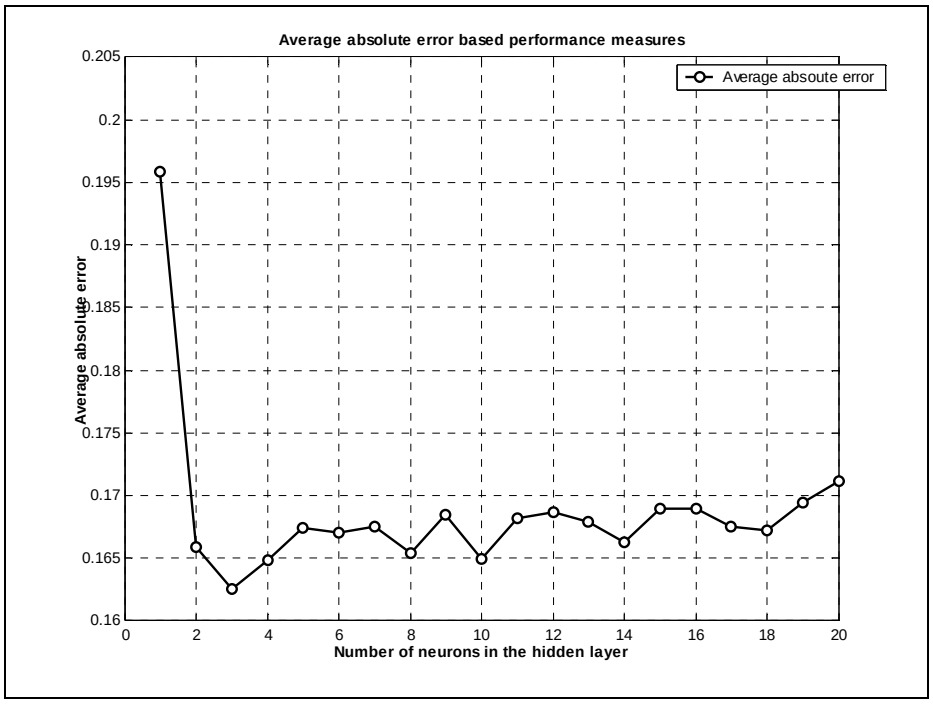

Figure 4-68 Average absolute error based performance measures, *Based on all input features*

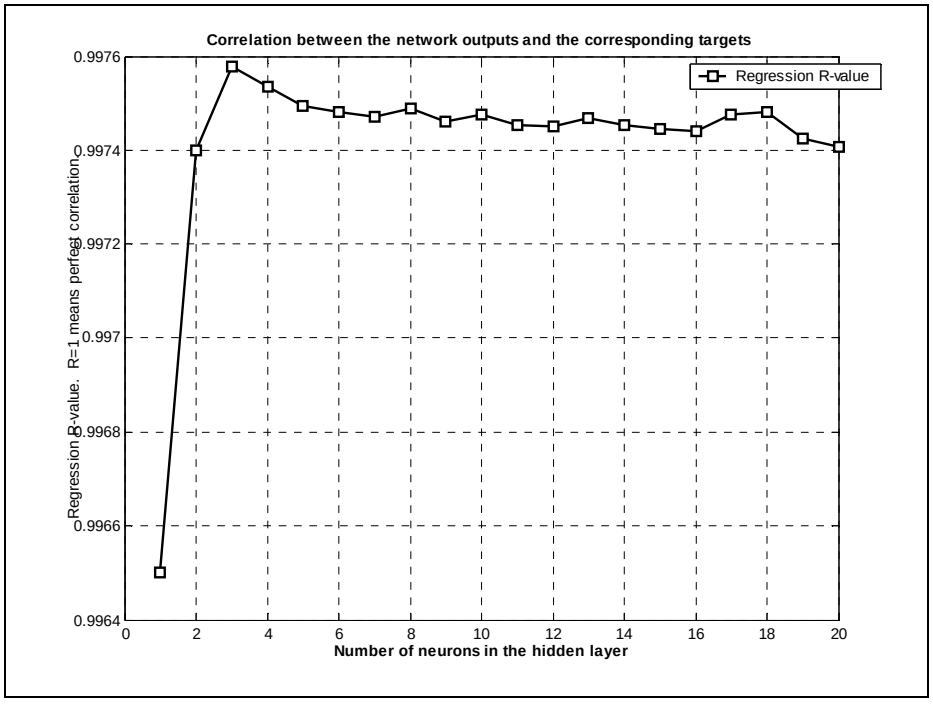

Figure 4-69 Regression R-value based performance measures, *Using all inputs*

As can be seen from figure 4-68 and 4-69 the optimum neural network is the 10-3-1 topology. A further analysis on using this network topology is shown in figures below.

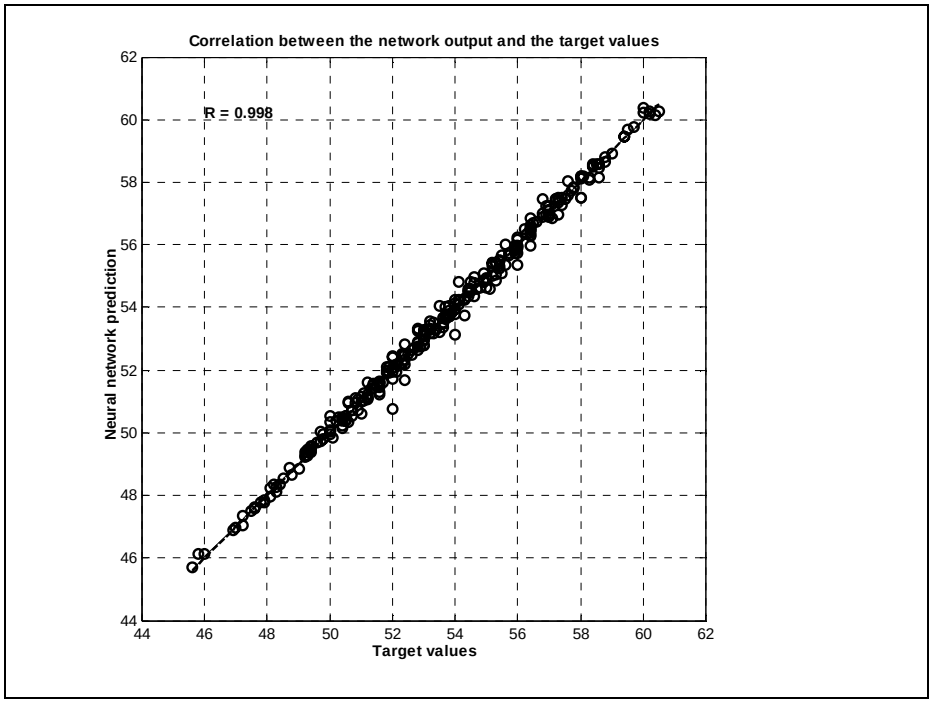

Figure 4-70 Correlation between the network prediction and the target values using the proposed model, *Using all inputs*

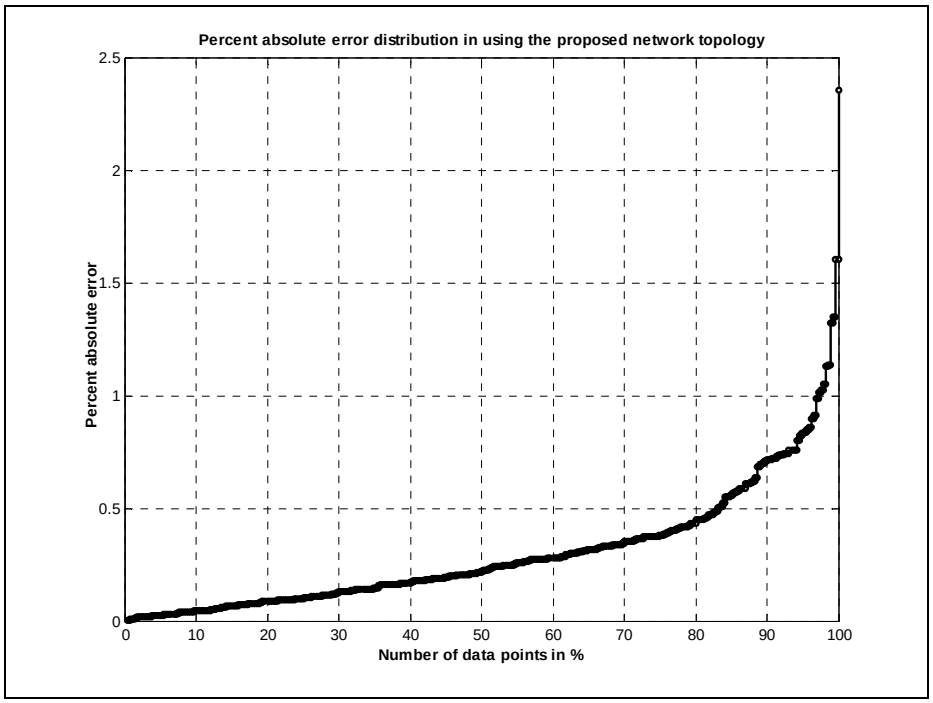

Figure 4-71 Distribution of percent absolute error in using the proposed input-output neural model, *Using all inputs*

The maximum percent absolute error in using the proposed neural network model is equal to 2.35% and 99% of the total prediction were under a percent absolute error of 1.5%. The result obtained shows an excellent model in all performance measuring criterion. The average absolute error in using this network model is equal to 0.163 units of CO_2 effluent level and the regression R-value is equal to 0.998.

It is also usual to predict the next value in a series from a fixed number of previous values (looking ahead a single time step). When the next value in a series is generated, further values can be estimated by feeding the newly-estimated value back into the network together with other previous values. The ability of the model to predict the next value of CO_2 effluent level using its own previous predictions is a very important point. This ability was analyzed and the result is shown in Figure 4-72.

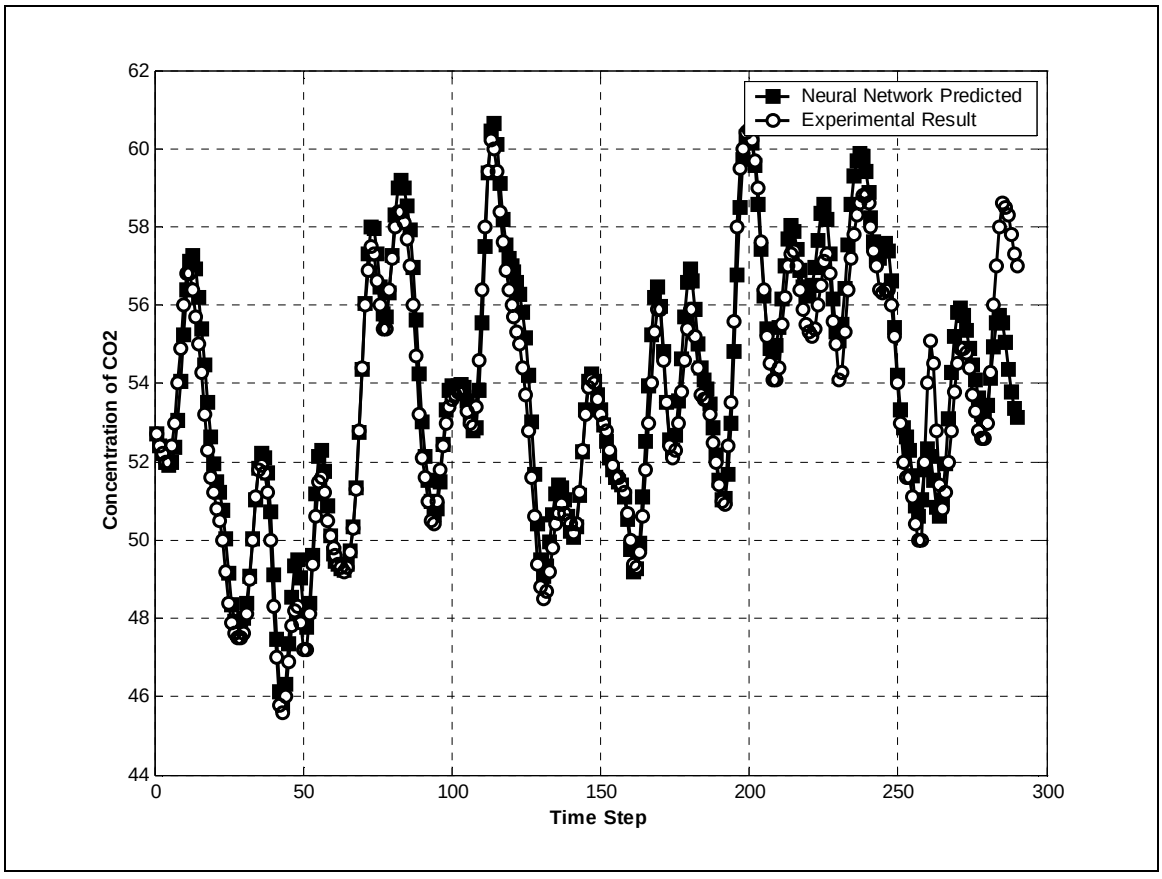

Figure 4-72 The ability of the model to predict the next value of CO_2 effluent level using its own previous predictions, *Based on all input features*

4.6.3 Neural network model after feature extraction

The Belue-Bauer method of selecting features is used to get the best set of features to train the network. The bar graph shown below gives the result obtained by applying this method.

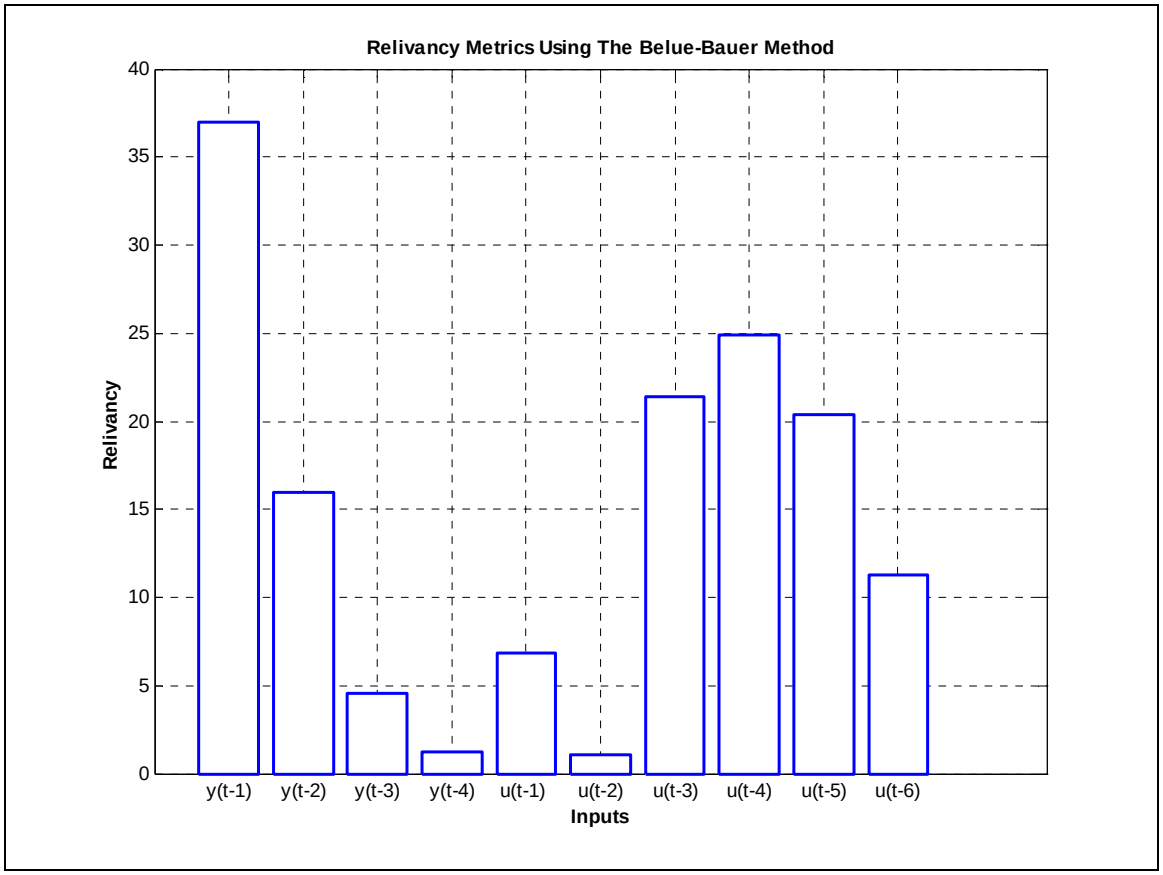

Figure 4-73 Result of the Belue-Bauer method of selecting features, *applied of CO_2 analyzer*

The decision to chose the features is guided by their relevancy value. Two or more alternatives may be taken and network performance may be used to arrive at the best alternative. Here we use the result of the Belue-Bauer method to eliminate some of the irrelevant features. As can be seen from the graph, {y(t-1), u(t-4)} looks more important features than the others. The result obtained by taking these as input is shown below.

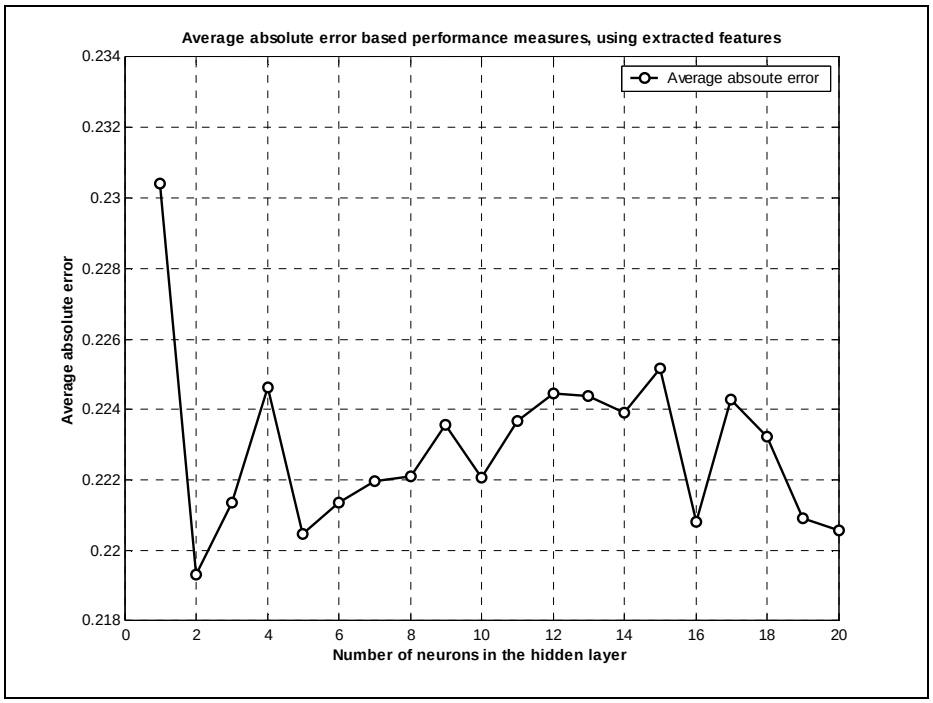

Figure 4-74 Average absolute error based performance measures, *Inputs are {y(t-1), u(t-4)}, CO_2 analyzer*

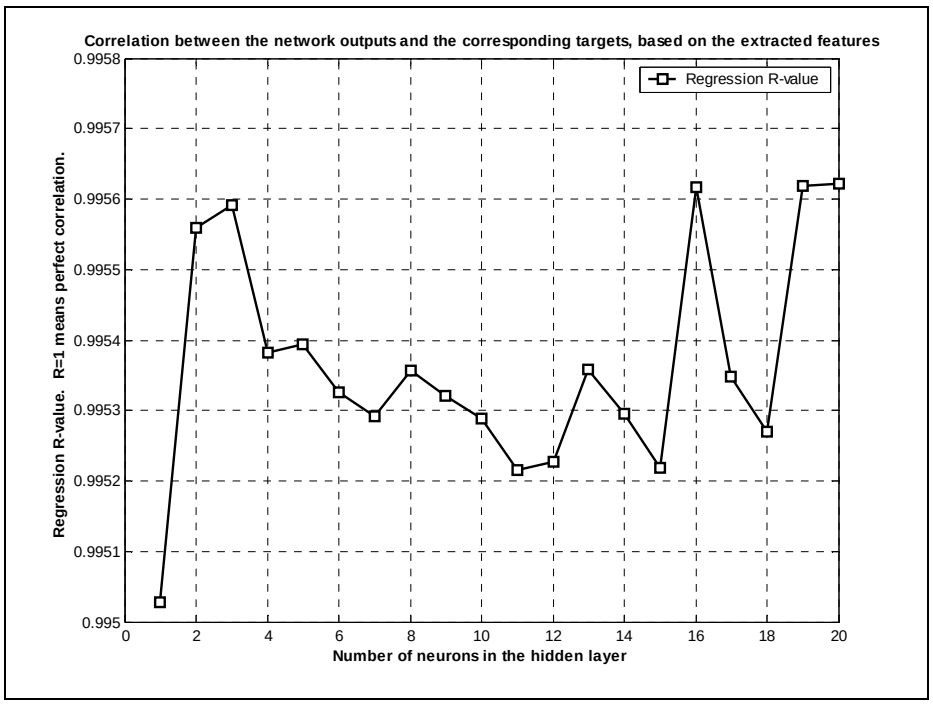

Figure 4-75 Regression R-value based performance measures, *Inputs are {y(t-1), u(t-4)}, CO_2 analyzer*

Based on the above two figures we suppose 2-2-1 neural topology to be the optimum. This is really a reduced and simplified model obtained from the application of the feature extraction method.

The average absolute error in using this model to predict the feature value of the CO_2 effluent level based on the input gas flaw rate at 4 time step behind and the CO_2 effluents at one time step behind is equal to 0.2426 Units of CO_2 effluent level, which when compared with 0.163 obtained for the previous model show performance loss due to feature extraction. But the need for small number of measured parameters may make the network model based on the extracted features more attractive.

The maximum percent absolute error in using this reduced model is equal to 3.5%. 95% of the predictions were made with percent absolute error less than 1.5% and the regression R- value is equal to 0.994.

The ability of the model to predict the next value of CO_2 effluent level using its own previous predictions was analyzed and the result is shown in Figure 4-76. The simplified and reduced neural network model obtained by applying the feature extraction method gives comparable performance results as the model base on all input features. This result show that, attempt to reduce the input features using feature extraction methods is worth to consider whenever possible.

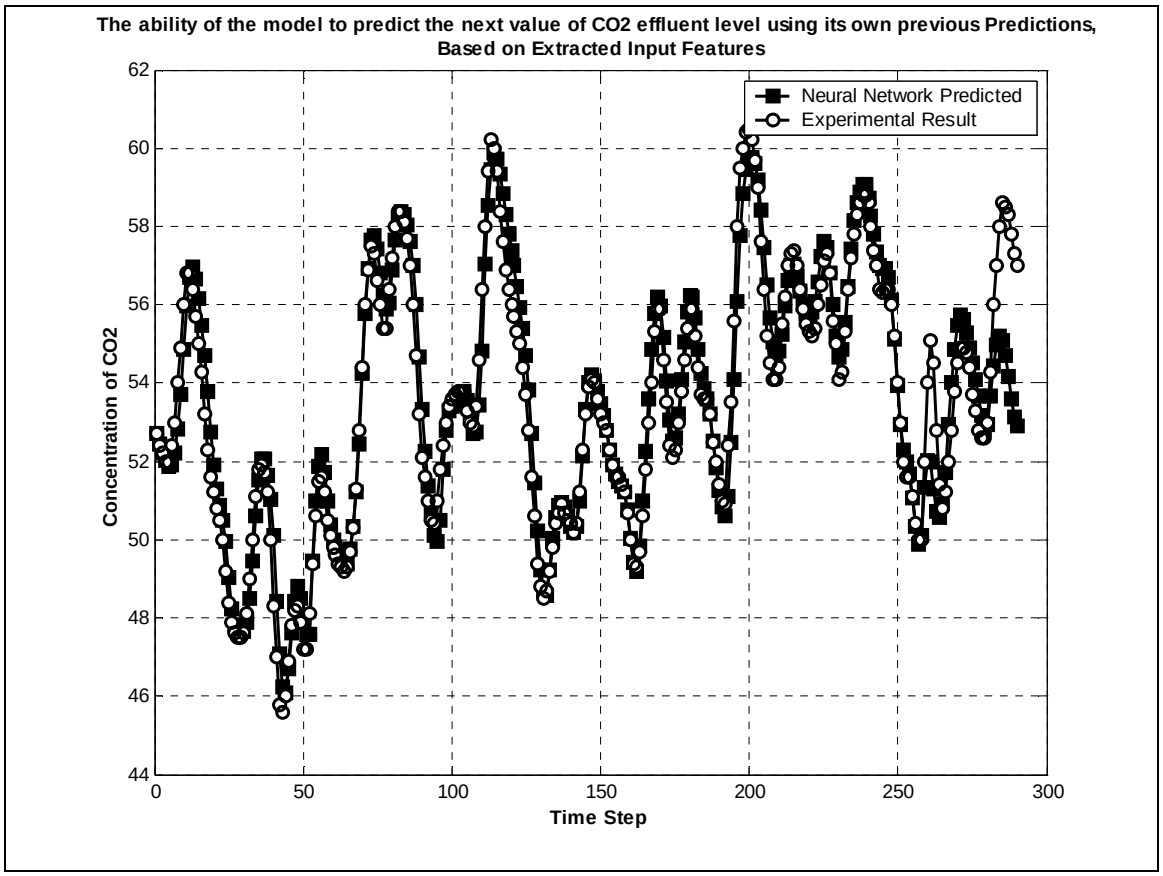

Figure 4-76 The ability of the model to predict the next value of CO_2 effluent level using its own previous predictions, *Based on extracted input features*

5 CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

ANNs are a rapidly growing facet of artificial intelligence, using a collection of simple processing units that are massively interconnected in order to produce meaningful behavior. This work presented a step by step procedure for developing a neural network model of five chemical engineering process. The benefits of such a technique include a better understanding of the final model, ease in model design (the topology and weight values are determined by way of comparing a set of different neural network models under different conditions), the generic nature of the model, and the graphical outputs help to visually analyze the nature of error distributions.

Neural network models were developed and employed to predict the vapor-liquid equilibrium (VLE) data for twelve different binary systems having different chemical structures and solution types (azeotrope-nonazeotrope) in various conditions (isothermal or isobaric). It is observed that the data found by neural network model gives an excellent agreement with the experimental data. In fact, the neural network model can be treated as a powerful means for VLE data prediction in a fast and reliable way, compared to the conventional thermodynamic based models.

The developed models were designed to calculate the four usually required VLE calculations. The models give excellent outputs for Bubble-T and Dew-T calculations almost for all binary mixtures considered in this thesis. Bubble-P and Dew-P calculation outputs were relatively poor. This is due to the nature of the collected VLE data. Different data preprocessing techniques were tried but didn't improve the performance. Collecting a good set of data that cover the pressure range in a smooth manner is essential.

The model developed for group of organic binary mixtures show excellent result too. The Belue-Bauer feature extraction method is used and reduces the number of pure

component properties required as input to the neural network model. Using accentricity, ω and critical volume, Vc of the first (*volatile*) component with the critical pressure, Pc and normal boiling temperature T_N of the second (*less volatile*) component the developed model predict the bubble pressure and the bubble temperature with good accuracy. The easiness of using the neural network model over the existing thermodynamic based models may make this approach attractive.

The results obtained in this thesis work showed that, conventionally used mathematically complex knowledge based thermodynamic modeling techniques can possibly be simplified by applying artificial neural network modeling techniques.

This study has also demonstrated the feasibility of using a neural network to capture the nonlinear and interacting relationships between the moisture content and different drying conditions of potato. The neural network trained with a limited number of data points was capable of predicting fresh data that was not used to train the network. The result obtained from the developed model can be used to analyze the drying characteristics of the product. The analysis results obtained are physically sound as expected from drying experience. Neural network modeling technique is really a purely data driven technique, which do not require knowledge on the functional relationships between variables.

The possibility of using artificial neural networks in modeling the temperature-time profile of an adiabatic reaction is also promising. The neural network predictor gives a consistent result for some part of starting temperature ranges. The early (induction), intermediate (acceleratory), and late (decay) periods of the adiabatic reactions are well followed. The inconsistency seen in using the neural network model to predict the temperature-time curve of the adiabatic reactions for starting temperatures above 70^oC indicates the need for additional adiabatic data at different starting temperatures. The result show that the three adiabatic data collected for the three starting temperatures are not enough to expose the neural network to different cases during the training phase.

Simulating a human operator controlling a chemical plant is also a good case where the advantage of using artificial neural networks is shown in this thesis. There are many data

like the one used in the problem in most chemical processing plants. A tool like artificial neural networks may be used to glean out the rules and regulations hidden in the data by way of mapping inputs and outputs of this kind.

In addition a neural network was used as process analyzer for a case study of CO_2 analyzer. It was proved that MLP-type network of a relatively simple structure made it possible to predict the CO_2 effluent from a furnace. Taking in to account difficulties in experimental conditions, complicated measurement equipment and unavoidable errors of devices used which limit the precision of laboratory measurement results, the accuracy of the results generated by the developed networks may be considered satisfactory for engineering calculations.

It is worth remembering that the accuracy of calculation results obtained from artificial neural networks are closely related to the accuracy and range of experimental data used for the network learning and testing.

5.2 Recommendations

Application of a neural network is a very attractive tool. It is simple, more cost effective, generic in nature, and more easily used, particularly by process plant engineers. The results presented demonstrate that neural networks have considerable potential in the field of data analysis, mainly because they require much less knowledge of the underlying physicochemical process.

In the hard sciences, neural network awareness and use has surged in recent years. This is due to several factors. Personal computers are now powerful enough to run neural network programs. Additionally, neural networks can overcome many of the shortcomings of traditional regression techniques, analyzing noisy data, incomplete data, and data with outliers. With all the advantages offered by these programs, why haven't neural networks been used by chemical engineers working in our country?

Considering results obtained in the thesis, available resources and technical feasibility, the following recommendations are suggested.

- The study in the thesis clearly shows the potential of using artificial neural networks in different chemical engineering modeling problems. Hence, creating awareness of neural network technology both in the graduate and undergraduate levels is a good action to consider.
- Investigating the application of neural networks for process modeling and condition monitoring in Ethiopian industrial context should be given special emphasis. There are several opportunities where neural networks can be considered in different industries, some of them are listed below.
    - Fault detection and diagnosis in cement process industries. The problem of clogging in the cyclone system is the major factor in creating failure. Here neural networks can be used to capture signs of imminent failure contained in the available process data.

- In polymer extrusion process neural networks can be developed to make an inferential estimator capable of predicting the viscosity of the polymer product based on different process parameters like the speed of the screw, temperature of the heating barrel and the like. Then incorporate this model within an automatic control system for the process
- The operation of rapid gravity filtration processes are another area where neural networks can give its potential. Model based on parameters like inlet turbidity, coagulant dosage, water inlet flow rate, differential pressure across filter, time since last filter clean etc. can be used as input to the model and filter run length and outlet turbidity can be predicted as the outlet from the model.
- ANNs can be used as O_2 analyzer to predict O_2 contents in a boiler effluent. The main idea here is that to infer the O_2 content in the flue gas from easily measured process variables. Process variables like fuel gas-burner pressure, outlet steam flow, combustion air flow and fuel gas flow can be used as input to the model.
- Artificial neural network technology is also applicable to many batch and continuous processes. These processes include food drying, fermentation, bio-reactions found in the bio-technology industries, energy management of city water distribution

Chemical engineers will find in this thesis a straightforward, clear, and practical beginning to artificial neural networks (ANNs) as it applies to their field. The thesis addressed various problems in a chemical engineering context. This thesis will also be of interest to graduate students, engineers practicing in industry, and computer scientists interested in chemical engineering applications.

6 APPENDICES

6.1 Summary of neural network models developed for the different binary mixtures

Table 6-1 Results obtained in Bubble-P calculation

NO.	MIXTURES	NUMBER OF DATA	TEMP. RANGE [K]	PRE. RANGE [Kpa]	NETWORK TOPOLOGY	REG. R-VA Y	AVE. ABS. ERROR Y	REG. R-VA P	AVE. ABS. ERROR P
1	Acetone-Chloroform	170	288.15 to 433.15	13.50 to 1376.10	2-7-2	1.0000	0.0047	1.0000	1.453
2	Acetone-Methanol	225	281.15 to 473.15	12.80 to 3985.40	2-10-2	0.9990	0.0070	1.0000	7.3568
3	Acetone-Water	146	293.15 to 523.15	2.30 to 101.33	2-5-2	0.9950	0.0169	0.9960	1.7503
4	Chloroform-Methanol	144	298.15 to 337.20	17.07 to 108.00	2-8-2	0.9990	0.0100	0.9990	0.6000
5	Ethanol-Benzene	194	293.15 to 462.05	8.40 to 1480.30	2-12-2	0.9980	0.0080	1.0000	2.9300
6	Ethanol-Water	821	294.95 to 623.15	5.00 to 18975	2-9-2	0.9970	0.0100	1.0000	22.5000
7	Methanol-Benzene	338	298.15 to 493.15	19.70 to 5764.4	2-10-2	0.9970	0.0097	1.0000	7.2400
8	Methanol-Water	400	285.25 to 523.15	3.60 to 7061.20	2-8-2	0.9970	0.0100	0.9960	50.0000
9	Methyl acetate-Methanol	292	293.15 to 413.39	15.60 to 1172.40	2-11-2	0.9990	0.0062	0.9990	2.8400
10	Methyl acetate-Water	74	298.15 to 429.51	3.20 to 1172.40	2-9-2	0.9910	0.0170	0.9990	10.1500
11	1-Propanol-Water	282	298.15 to 372.25	2.93 to 110.13	2-9-2	0.9980	0.0104	1.0000	0.6890
12	2-Propanol-Water	364	303.15 to 573.15	4.00 to 12349	2-8-2	0.9960	0.0120	1.0000	0.4100

Table 6-2 Results obtained in Bubble-T calculation

NO.	MIXTURES	NUMBER OF DATA	TEMP. RANGE [K]	PRE. RANGE [Kpa]	NETWORK TOPOLOGY	REG. R-VAY	AVE. ABS. ERROR Y	REG. R-VAP	AVE. ABS. ERROR P
1	Acetone-Chloroform	170	288.15 to 433.15	13.50 to 1376.10	2-6-2	1.0000	0.0050	1.0000	0.6800
2	Acetone-Methanol	225	281.15 to 473.15	12.80 to 3985.40	2-7-2	0.9990	0.0100	1.0000	0.5000
3	Acetone-Water	146	293.15 to 523.15	2.30 to 101.33	2-7-2	0.9970	0.0140	0.9990	0.7000
4	Chloroform-Methanol	144	298.15 to 337.20	17.07 to 108.00	2-5-2	0.9986	0.0100	0.9995	0.2370
5	Ethanol-Benzene	194	293.15 to 462.05	8.40 to 1480.30	2-7-2	0.9978	0.0085	0.9997	0.6520
6	Ethanol-Water	821	294.95 to 623.15	5.00 to 18975	2-7-2	0.9950	0.0152	0.9990	1.8417
7	Methanol-Benzene	338	298.15 to 493.15	19.70 to 5764.4	2-10-2	0.9960	0.0103	1.0000	0.7440
8	Methanol-Water	400	285.25 to 523.15	3.60 to 7061.20	2-7-2	0.9980	0.0121	0.9950	3.7082
9	Methyl acetate-Methanol	292	293.15 to 413.39	15.60 to 1172.40	2-8-2	0.9990	0.0066	1.0000	0.3509
10	Methyl acetate-Water	74	298.15 to 429.51	3.20 to 1172.40	2-11-2	0.9910	0.0156	0.9980	1.1781
11	1-Propanol-Water	282	298.15 to 372.25	2.93 to 110.13	2-8-2	0.9980	0.0106	1.0000	0.3725
12	2-Propanol-Water	364	303.15 to 573.15	4.00 to 12349	2-6-2	0.9960	0.0123	0.9990	2.6708

Table 6-3 Results obtained in Dew-P calculation

NO.	MIXTURES	NUMBER OF DATA	TEMP. RANGE [K]	PRE. RANGE [Kpa]	NETWORK TOPOLOGY	REG. R-VA Y	AVE. ABS. ERROR Y	REG. R-VA P	AVE. ABS. ERROR P
1	Acetone-Chloroform	170	288.15 to 433.15	13.50 to 1376.10	2-8-2	1.0000	0.0049	1.0000	3.0191
2	Acetone-Methanol	225	281.15 to 473.15	12.80 to 3985.40	2-8-2	0.9990	0.0114	1.0000	10.9587
3	Acetone-Water	146	293.15 to 523.15	2.30 to 101.33	2-13-2	0.9800	0.0412	1.0000	17.7910
4	Chloroform-Methanol	144	298.15 to 337.20	17.07 to 108.00	2-9-2	0.9980	0.0136	0.9990	0.7301
5	Ethanol-Benzene	194	293.15 to 462.05	8.40 to 1480.30	2-8-2	0.9990	0.0116	1.0000	5.0784
6	Ethanol-Water	821	294.95 to 623.15	5.00 to 18975	2-11-2	0.9940	0.0172	1.0000	37.7490
7	Methanol-Benzene	338	298.15 to 493.15	19.70 to 5764.4	2-13-2	0.9960	0.0174	1.0000	11.4377
8	Methanol-Water	400	285.25 to 523.15	3.60 to 7061.20	2-10-2	0.9980	0.0140	0.9990	32.1820
9	Methyl acetate-Methanol	292	293.15 to 413.39	15.60 to 1172.40	2-8-2	0.9990	0.0075	1.0000	3.3496
10	Methyl acetate-Water	74	298.15 to 429.51	3.20 to 1172.40	2-10-2	0.9170	0.1190	0.9950	21.6475
11	1-Propanol-Water	282	298.15 to 372.25	2.93 to 110.13	2-10-2	0.9960	0.0196	1.0000	0.6570
12	2-Propanol-Water	364	303.15 to 573.15	4.00 to 12349	2-11-2	0.9960	0.0176	0.9990	22.6377

Table 6-4 Results obtained in Dew-T calculation

NO.	MIXTURES	NUMBER OF DATA	TEMP. RANGE [K]	PRE. RANGE [Kpa]	NETWORK TOPOLOGY	REG. R-VAY	AVE. ABS. ERRORY	REG. R-VAP	AVE. ABS. ERRORP
1	Acetone-Chloroform	170	288.15 to 433.15	13.50 to 1376.10	2-9-2	0.9990	0.0050	1.0000	0.2600
2	Acetone-Methanol	225	281.15 to 473.15	12.80 to 3985.40	2-8-2	0.9990	0.0110	1.0000	0.5690
3	Acetone-Water	146	293.15 to 523.15	2.30 to 101.33	2-10-2	0.9860	0.0387	0.9970	1.0098
4	Chloroform-Methanol	144	298.15 to 337.20	17.07 to 108.00	2-6-2	0.9980	0.0143	0.9990	0.2671
5	Ethanol-Benzene	194	293.15 to 462.05	8.40 to 1480.30	2-7-2	0.9970	0.0177	1.0000	0.7671
6	Ethanol-Water	821	294.95 to 623.15	5.00 to 18975	2-6-2	0.9930	0.0212	0.9980	3.4428
7	Methanol-Benzene	338	298.15 to 493.15	19.70 to 5764.4	2-14-2	0.9960	0.0179	1.0000	0.6862
8	Methanol-Water	400	285.25 to 523.15	3.60 to 7061.20	2-11-2	0.9980	0.0143	0.9960	3.2300
9	Methyl acetate-Methanol	292	293.15 to 413.39	15.60 to 1172.40	2-9-2	0.9990	0.0078	1.0000	0.3781
10	Methyl acetate-Water	74	298.15 to 429.51	3.20 to 1172.40	2-11-2	0.9120	0.1246	0.9930	3.0404
11	1-Propanol-Water	282	298.15 to 372.25	2.93 to 110.13	2-8-2	0.9960	0.0213	0.4831	0.9990
12	2-Propanol-Water	364	303.15 to 573.15	4.00 to 12349	2-9-2	0.9960	0.0188	0.9990	1.7498

6.2 MATLAB files to construct ANNs

6.2.1 Guide to use the computer programs

Data collection:

Appropriate data for a given problem is first collected and put in a separate data file, like, **VLED.txt**. Here if data are collected from different sources we should give emphasis to unit consistencies and different other data integration issues.

Network objects construction and model selection:

Here a set of competing neural network models are constructed. This step is used to compare models with different number of neurons in their hidden layer under various neural network settings. Here we can experiment on the effect of different network parameters like:

- Data transformation.
- Transfer function in the hidden layer.
- Transfer function in the output layer.
- Training algorithm.
- Parameters of the given training algorithm.

Comparing the performance of models under different settings is used to pick out the optimum neural network topology plus the optimum neural network settings. COMPARE.m is the MATLAB script file to do such work. This program is given in section 6.2.2. The main activities in the program are summarized bellow:

- Load the raw data into the workspace and put it in to the appropriate input/output array format.
- Data pre-processing step is then followed. This step includes data transformation techniques. Linear transformation is used for all problems in this thesis work. This step transform the raw data in a range between [β, $1-\beta$], because MLPs do not train properly on 0s and 1s as inputs [12]. The linear transformation is usually used as shown in Equation (3.1).

So that $L_n(x_{\min(n)})=\beta$ and $L_n(x_{\max(n)})=1-\beta$. Checking network performances at different values of β is frequently considered.

- The pre-processed data is then divided into three disjoint data sets, training sets, testing sets and validation sets.
- Set of competing network topologies is then constructed. Here twenty neural network models differing in the number of neuron in their hidden layers are constructed. Types of activation function to be used in the hidden and output layers are set here.
- Network training parameters are set. This parameters are
    - Network training performance functions
        1. Mean squared error
        2. Mean absolute error
        3. Mean squared error with regularization
    - Network training algorithm. Choose one training algorithm from the many available backpropagation techniques.
        1. Gradient descent backpropagation
        2. Gradient descent with adaptive learning rate backpropagation :
        3. Gradient descent with momentum & adaptive learning rate backpropagation:
        4. Levenberg-Marquardt backpropagation:
    - Set parameters appropriate for the selected training algorithm for example for Levenberg-Marquardt backpropagation, which is used thought this thesis work due to its good performance, we need to set the following parameters.
        1. Maximum number of epochs to train
        2. Performance goal
        3. Maximum validation failures
        4. Factor to use for memory/speed trade off.
        5. Minimum performance gradient
        6. Initial Mu

7. Mu decrease factor
8. Mu increase factor
9. Maximum Mu

- Each neural network topology is trained under a given training parameters and its performance is measured.
- Graphical outputs showing the performance of the network models is used to select the best network topology.

Analysis of the performance of the selected model and extracting the parameters:

Here the performance of the selected model is further analyzed based on different performance measuring functions. If results obtained are satisfactory the network descriptions are extracted and saved in a separate file. EXTRACT.m is the MATLAB script file to do such work.

Put the selected neural network model in its operating mode:

Here the parameters of the best model obtained from EXTRACT.m are used to put the neural network in operating mode. A program called CONSTRUCT.m which is a MATLAB function file put the neural network model in its operating mode.

Use the neural network model for the required purpose:

This is a program written to use the neural network model constructed in CONSTRUCT.m for the analysis work that we like to put our model.

Feature extraction:

In some of the problems sited in this thesis work the advantage of applying a feature extraction technique is analyzed. The Belue-Bauer method of selecting features is simulated in a MATLAB program and used to eliminate irrelevant features for a given problem. RELEVANCY.m is a MATLAB script file to perform the Belue-Bauer method of selecting features.

6.2.2 COMPARE.m

```
clear
% Load the raw data from the appropriate file
load VLED.txt;
rp=VLED(:,1);
p=rp';
rx=VLED(:,2);
x=rx';
ry=VLED(:,3);
y=ry';
rt=VLED(:,4);
t=rt';

% This is the data transformation steps. this step normalize each data vector so that
%they  are between eee and (1-eee)
mmp=minmax(p);
mmx=minmax(x);
mmy=minmax(y);
mmt=minmax(t);

eee=0.15;
pp=(1 - 2 * eee) * ((p - mmp(1))/ (mmp(2) - mmp(1))) + eee;
px=(1 - 2 * eee) * ((x - mmx(1))/ (mmx(2) - mmx(1))) + eee;
py=(1 - 2 * eee) * ((y - mmy(1))/ (mmy(2) - mmy(1))) + eee;
pt=(1 - 2 * eee) * ((t - mmt(1))/ (mmt(2) - mmt(1))) + eee;

% Put the pre-processed output vectors in array form.
out(1,:)=py;
out(2,:)=pp;

% Put the pre-processed input vectors in array form.
in(1,:)=pt;
in(2,:)=px;
% This step separate the total data into three separate data sets for training, validation
% and tesiting
 [R,Q] = size(x);
iitst = 2:4:Q;
iival = 4:4:Q;
iitr = [1:4:Q 3:4:Q];
val.P=in(:,iival); val.T=out(:,iival);
test.P=in(:,iitst); test.T=out(:,iitst);
ptr = in(:,iitr); ttr = out(:,iitr);

% Construct set of neural networks with number of hidden neurons varying between "hni"
% and "hnf" at "del" step
hni=1;
hnf=20;
del=1;
for hn=hni:del:hnf;

   % Here the neural network is constructed
```

```
net = network;
net.numInputs = 1;
net.inputs{1}.size = 2;
net.numLayers = 2;
net.layers{1}.size =hn;
net.layers{2}.size = 2;
net.inputConnect(1) = 1;
net.layerConnect(2, 1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'purelin';
net.biasConnect = [1; 1];

% The network performance measuring function and the training algorithms are set
%  here
net.performFcn = 'mse';
net.trainFcn = 'trainlm';

% Training parameters for the training algorithm used above are manipulated here
net.trainParam.epochs= 500;
net.trainParam.goal=        0;
net.trainParam.max_fail=    100;
net.trainParam.mem_reduc=   1;
net.trainParam.min_grad= 1e-10;
net.trainParam.mu =     0.001;
net.trainParam.mu_dec=    0.1;
net.trainParam.mu_inc  =   10;
net.trainParam.mu_max=    1e10;
net.trainParam.show =      1;
% net.trainParam.time =     1200;

% This call a training procedure
   [net,tr]=train(net,ptr,ttr,[],[],val,test);

% This use the trained neural network to predict output values for the entire data
an=sim(net,in);

% This is the data post-processing step
pouty = ((an(1,:) - eee) * (mmy(2) - mmy(1)) / (1 - 2 * eee)) + mmy(1);
poutp = ((an(2,:) - eee) * (mmp(2) - mmp(1)) / (1 - 2 * eee)) + mmp(1);

% This calculate and store the percent absolute error for each network model
eey(hn,:)=((py- an(1,:))./py)*100;
eep(hn,:)=((p-poutp)./p)*100;

% This calculate and store the average absolute error for each network model
dely(hn,:)=sum(abs(y-pouty))/length(x);
delp(hn,:)=sum(abs(p-poutp))/length(x);

% This perform a correlation analysis between the network prediction and target
```

```
 % values
   [yy1(hn),yy2(hn),yy3(hn)]=postreg(pouty,y);
   [pp1(hn),pp2(hn),pp3(hn)]=postreg(poutp,p);

end

% This is graphical output to show the performance of the different networks based on
%the  regression coefficient
figure(1)
plot(hni:del:hnf,yy3(hni:del:hnf),'bo-');
k=legend('Bubble composition');
title('Correlation analysis between network predictions and the corresponding targets (Type
BUBL P: Calculate {yi} and P, given {xi} and T) ');
xlabel('Number of neurons in the hidden layer');
ylabel(' Regression R-value.  R=1 means perfect correlation.');
grid;
zoom;

figure(2)
plot(hni:del:hnf,pp3(hni:del:hnf),'r*-');
k=legend('Bubble presure');
title('Correlation analysis between network predictions and the corresponding targets (Type
BUBL P: Calculate {yi} and P, given {xi} and T)');
xlabel('Number of neurons in the hidden layer');
ylabel('Regression R-value.  R=1 means perfect correlation.');
grid;
zoom;

% This is graphical output to show the performance of the different networks based on
%the  average absolute error.
figure(3)
plot(hni:del:hnf,dely(hni:del:hnf),'bo-');
k=legend('Bubble composition');
title('Average absolute error (Type BUBL P: Calculate {yi} and P, given {xi} and T)');
xlabel('Number of neurons in the hidden layer');
ylabel('Average absolute error');
grid;
zoom;

figure(4)
plot(hni:del:hnf,delp(hni:del:hnf),'r*-');
k=legend('Bubble Pressure.');
title('Average absolute error (Type BUBL P: Calculate {yi} and P, given {xi} and T)');
xlabel('Number of neurons in the hidden layer');
ylabel('Average absolute error');
grid;
zoom;

%End program
```

6.2.3 EXTRACT.m

```
clear
% Load the raw data from the appropriate file
load VLED.txt;
rp=VLED(:,1);
p=rp';
rx=VLED(:,2);
x=rx';
ry=VLED(:,3);
y=ry';
rt=VLED(:,4);
t=rt';

% Thid step will determine the minimum and maximum value of each vector
mmp=minmax(p);
mmx=minmax(x);
mmy=minmax(y);
mmt=minmax(t);

% This is the data transformation steps. this step normalize each data vector so that %they
are between eee and (1-eee)
eee=0.15;
pp=(1 - 2 * eee) * ((p - mmp(1))/ (mmp(2) - mmp(1))) + eee;
px=(1 - 2 * eee) * ((x - mmx(1))/ (mmx(2) - mmx(1))) + eee;
py=(1 - 2 * eee) * ((y - mmy(1))/ (mmy(2) - mmy(1))) + eee;
pt=(1 - 2 * eee) * ((t - mmt(1))/ (mmt(2) - mmt(1))) + eee;

% Put the pre-processed output(target) vectors in a proper format for network training
out(1,:)=py;
out(2,:)=pp;

% Put the pre-processed input vectors in a proper format for network training
in(1,:)=px;
in(2,:)=pt;

% This step separate the total data into three separate data sets for training, validation
% and tesiting
 [R,Q] = size(x);
iitst = 2:4:Q;
iival = 4:4:Q;
iitr = [1:4:Q 3:4:Q];
val.P=in(:,iival); val.T=out(:,iival);
test.P=in(:,iitst); test.T=out(:,iitst);
ptr = in(:,iitr); ttr = out(:,iitr);
% Here the neural network is constructed
net = network;
net.numInputs = 1;
net.inputs{1}.size = 2;
net.numLayers = 2;
net.layers{1}.size =7;
net.layers{2}.size = 2;
```

```
net.inputConnect(1) = 1;
net.layerConnect(2, 1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'purelin';
net.biasConnect = [1; 1];

% The network performance measuring function and the training algorithms are set %here
net.performFcn = 'mse';
net.trainFcn = 'trainlm';

% Training parameters for the training algorithm used above are manipulated here
net.trainParam.epochs= 500;
net.trainParam.goal=        0;
net.trainParam.max_fail=    100;
net.trainParam.mem_reduc=   1;
net.trainParam.min_grad= 1e-10;
net.trainParam.mu =     0.001;
net.trainParam.mu_dec=    0.1;
net.trainParam.mu_inc  =   10;
net.trainParam.mu_max=   1e10;
net.trainParam.show =      1;
% net.trainParam.time =     1200;

% This call a training procedure
[net,tr]=train(net,ptr,ttr,[],[],val,test);

% This use the trained neural network to predict output values for the entire data
an=sim(net,in);

% This is the data post-processing step
pouty = ((an(1,:) - eee) * (mmy(2) - mmy(1)) / (1 - 2 * eee)) + mmy(1);
poutp = ((an(2,:) - eee) * (mmp(2) - mmp(1)) / (1 - 2 * eee)) + mmp(1);

% This calculate the percent absolute error for each network model
eey=((py- an(1,:))./py)*100;
eep=((p-poutp)./p)*100;
soy=sort(abs(eey));
sop=sort(abs(eep));
st=length(eey);
stt=1:1:st;
xc=(stt*100)/st;

figure(1)
clf
plot(stt,eey);
title('Percent Errors in Predicting Bubble Compositions');
ylabel('Percent Error');
xlabel('Data Points');
grid;
```

```
zoom;

figure(2)
clf
plot(stt,eep);
title('Percent Errors in Predicting Bubble Pressures');
ylabel('Percent Error');
xlabel('Data Points');
grid;
zoom;

figure(3)
clf
stairs(xc,soy,'b');
title('Percent  Absolute Error Distribution in Predicting the Bubble Comp., Sorted');
ylabel('Percent Absolute Error');
xlabel('Number of data points in Percent');
grid;
zoom;

figure(4)
clf
stairs(xc,sop,'b');
title('Percent Absolute Error Distribution in predicting the Bubble Pressure');
ylabel('Percent Absolute Error');
xlabel('Number of data points in Percent');
grid;
zoom;

% This calculate and store the average absolute error in using the proposed model
dely=sum(abs(y-pouty))/length(x);
delp=sum(abs(p-poutp))/length(x);

% This perform a Correlation analysis between the network prediction and target %values
figure(5)
clf
[yy1,yy2,yy3]=postreg(pouty,y);
title('Correlation Analysis Betweent Network Prediction and Target Values, Predicting the
Bubble Compositions');
xlabel('Target Value');
ylabel('Neural Network Prediction');
grid;
zoom;

figure(6);
clf
[pp1,pp2,pp3]=postreg(poutp,p);
title('Correlation Analysis Betweent Network Prediction and Target Values, Predicting the
Bubble Pressures');
xlabel('Target Value');
ylabel('Neural Network Prediction');
```

```
grid;
zoom;

% Extract weights and biases of the proposed network topology
W=net.IW{1,1};
b1=net.b{1};
U=net.LW{2,1};
b2=net.b{2};


%Save the generated hidden layer weights to a file
fid=fopen('hw.txt','w');
fprintf(fid,'%12.6g  %12.6g\n',W');
fclose(fid);

%Save the generated hidden layer biases to a file
fid=fopen('hb.txt','w');
fprintf(fid,'%12.6g\n',b1);
fclose(fid);

%Save the generated output layer weights to a file
fid=fopen('ow.txt','w');
fprintf(fid,'%12.6g %12.6g %12.6g %12.6g %12.6g %12.6g %12.6g\n',U');
fclose(fid);

%Save the generated output layer biases to a file
fid=fopen('ob.txt','w');
fprintf(fid,'%12.6g\n',b2);
fclose(fid);

% End of Program
```

6.2.4 CONSTRUCT.m

```
function f = CONSTRUCT(lc,et)
hn=7;
nn=2;
sx=0;
lx=1;
sy=0;
ly=1;
sp=8.4;
lp=1480.3;
st=293.1500;
lt=462.0500;
eee=0.15;
pc =(1 - 2 * eee) * ((lc - sx)/ (lx -sx)) + eee;
pt =(1 - 2 * eee) * ((et - st)/ (lt - st)) + eee;
x(1)= pc;
x(2)= pt;
load hw.txt;
weight1=hw(:,:);
load hb.txt;
bias1=hb(:,:);
load ow.txt;
weight2=ow(:,:);
load ob.txt;
bias2=ob(:,:);
for m=1:hn
   r(m)=0;
   for n=1:nn
      r(m)= r(m)+ weight1(m,n)*x(n);
   end
   r(m)=r(m)+bias1(m);
   yy(m)=logsig(r(m));
end
for j=1:2
   s(j)=0;
   for m=1:hn
      s(j)= s(j)+weight2(j,m)*yy(m);
   end
   s(j)=s(j)+bias2(j);
   z(j)=purelin(s(j));
end
ty = ((z(1) - eee) * (ly - sy) / (1 - 2 * eee)) + sy;
tp = ((z(2) - eee) * (lp - sp) / (1 - 2 * eee)) + sp;
f=[ty tp];

%END Function
```

6.2.5 USE.m

This MATLAB script file use the constructed functional form of the neural network model to do different analysis

```
int=300;
delt=10;
fit=320;
inx=0.1;
delx=0.01;
fix=0.9;
i=0;
R=length(inx:delx:fix)
C=length(int:delt:fit)
z=zeros(R,2*C);
for xx=inx:delx:fix;
   i=i+1;
   r=1;
   s=2;
   for tt=int:delt:fit
      z(i,r:s)=CONSTRUCT(xx,tt);
      r=s+1;
      s=r+1;
   end
end
figure(1)
clf
plot(inx:delx:fix,z(:,2:2:2*C),'ro--',z(:,1:2:2*C),z(:,2:2:2*C),'bs-');
legend('Sat.Liq = 300 [K]','Sat.Liq = 310','Sat.Liq = 320','Sat.Vap = 300','Sat.Vap =
310','Sat.Vap = 320');
xlabel('x and predicted y');
ylabel('Predicted Pressures [kpa]');
grid;
zoom;

%End program
```

6.2.6 RELIEVANCY.m

This MATLAB script program file applies the Belue–Bauer method of feature extraction method.

The bar graph output from this program show the relevancy metric results for each input features.

```
clear
% Load the raw data from the appropriate file
load VLED2.txt;
rx1=VLED2(:,1);
x1=rx1';

rx2=VLED2(:,2);
x2=rx2';

rx3=VLED2(:,3);
x3=rx3';

rx4=VLED2(:,4);
x4=rx4';

rx5=VLED2(:,5);
x5=rx5';

rx6=VLED2(:,6);
x6=rx6';

rx7=VLED2(:,7);
x7=rx7';

rx8=VLED2(:,8);
x8=rx8';

rx9=VLED2(:,9);
x9=rx9';

rx10=VLED2(:,10);
x10=rx10';

rx11=VLED2(:,11);
x11=rx11';

rx12=VLED2(:,12);
x12=rx12';

rx13=VLED2(:,13);
x13=rx13';

rx14=VLED2(:,14);
x14=rx14';
rx15=VLED2(:,15);
x15=rx15';
```

```
rx16=VLED2(:,16);
x16=rx16';

rx17=VLED2(:,17);
x17=rx17';

% Thid step will determine the minimum and maximum value of each vector
mmx1=minmax(x1);
mmx2=minmax(x2);
mmx3=minmax(x3);
mmx4=minmax(x4);
mmx5=minmax(x5);
mmx6=minmax(x6);
mmx7=minmax(x7);
mmx8=minmax(x8);
mmx9=minmax(x9);
mmx10=minmax(x10);
mmx11=minmax(x11);
mmx12=minmax(x12);
mmx13=minmax(x13);
mmx14=minmax(x14);
mmx15=minmax(x15);
mmx16=minmax(x16);
mmx17=minmax(x17);

% This is the data transformation steps. this step normalize each data vector so that they
% are between eee and (1-eee)
eee=0.15;
px1=(1 - 2 * eee) * ((x1 - mmx1(1))/ (mmx1(2) - mmx1(1))) + eee;
px2=(1 - 2 * eee) * ((x2 - mmx2(1))/ (mmx2(2) - mmx2(1))) + eee;
px3=(1 - 2 * eee) * ((x3 - mmx3(1))/ (mmx3(2) - mmx3(1))) + eee;
px4=(1 - 2 * eee) * ((x4 - mmx4(1))/ (mmx4(2) - mmx4(1))) + eee;
px5=(1 - 2 * eee) * ((x5 - mmx5(1))/ (mmx5(2) - mmx5(1))) + eee;
px6=(1 - 2 * eee) * ((x6 - mmx6(1))/ (mmx6(2) - mmx6(1))) + eee;
px7=(1 - 2 * eee) * ((x7 - mmx7(1))/ (mmx7(2) - mmx7(1))) + eee;
px8=(1 - 2 * eee) * ((x8 - mmx8(1))/ (mmx8(2) - mmx8(1))) + eee;
px9=(1 - 2 * eee) * ((x9 - mmx9(1))/ (mmx9(2) - mmx9(1))) + eee;
px10=(1 - 2 * eee) * ((x10 - mmx10(1))/ (mmx10(2) - mmx10(1))) + eee;
px11=(1 - 2 * eee) * ((x11 - mmx11(1))/ (mmx11(2) - mmx11(1))) + eee;
px12=(1 - 2 * eee) * ((x12 - mmx12(1))/ (mmx12(2) - mmx12(1))) + eee;
px13=(1 - 2 * eee) * ((x13 - mmx13(1))/ (mmx13(2) - mmx13(1))) + eee;
px14=(1 - 2 * eee) * ((x14 - mmx14(1))/ (mmx14(2) - mmx14(1))) + eee;
px15=(1 - 2 * eee) * ((x15 - mmx15(1))/ (mmx15(2) - mmx15(1))) + eee;
px16=(1 - 2 * eee) * ((x16 - mmx16(1))/ (mmx16(2) - mmx16(1))) + eee;
px17=(1 - 2 * eee) * ((x17 - mmx17(1))/ (mmx17(2) - mmx17(1))) + eee;


% Put the pre-processed output vectors in array form as py
out(1,:)=px16;
out(2,:)=px17;
```

```
% Put the pre-processed input vectors in a proper form
in(1,:)=px1;
in(2,:)=px2;
in(3,:)=px3;
in(4,:)=px4;
in(5,:)=px5;
in(6,:)=px6;
in(7,:)=px7;
in(8,:)=px8;
in(9,:)=px9;
in(10,:)=px10;
in(11,:)=px11;
in(12,:)=px12;
in(13,:)=px13;
in(14,:)=px14;
in(15,:)=px15;

[R,Q] = size(x16);
iitst = 2:4:Q;
iival = 4:4:Q;
iitr = [1:4:Q 3:4:Q];

val.P=in(:,iival); val.T=out(:,iival);
test.P=in(:,iitst); test.T=out(:,iitst);
ptr = in(:,iitr); ttr = out(:,iitr);
zzz=length(ptr);

ss =0;
epo=30;
for ii=1:epo

net = network;
net.numInputs = 1;
net.inputs{1}.size = 15;
net.numLayers = 2;
net.layers{1}.size =10;
net.layers{2}.size = 1;
net.inputConnect(1) = 1;
net.layerConnect(2, 1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'purelin';
net.biasConnect = [0; 0];

net.performFcn = 'mse';
net.trainFcn = 'trainlm';

net.trainParam.epochs= ii;
net.trainParam.goal=        0;
net.trainParam.max_fail=    1000;
```

```
net.trainParam.mem_reduc=    1;
net.trainParam.min_grad= 1e-10;
net.trainParam.mu =      0.001;
net.trainParam.mu_dec=     0.1;
net.trainParam.mu_inc  =    10;
net.trainParam.mu_max=    1e10;
net.trainParam.show =       1;
% net.trainParam.time =      1200;

net=train(net,in,out)

W=net.IW{1,1};
b1=net.b{1};
U=net.LW{2,1};
b2=net.b{2};

for nn=1:15;
   la(nn)=0;
  for mm=1:10;
     la(nn)=la(nn)+W(mm,nn)*W(mm,nn);
  end
end
ss =ss +la;
end
comp=ss/epo;
figure(1)
clf
bar(1:1:15,ss);
title('Results of the Belue-Bauer Method of Feature Extraction for the VLE data');
xlabel('Input features');
ylabel('Relivancy metric values');

zoom;
grid;
% End of Program
```

7 BIBLIOGRAPHY

[1]ANTONIO AUGUSTO GORNI, *Tutorial on Neural Networks,* Companihia Siderurgica Paulista-COSPIA, Brasil, http://www.gorni.eng.br/e/neurabst.html

[2]K.H.CHU, *A Neural Network Model for Prediction of Binary Metal Biosorption,* Department of Chemical and Process Engineering, University of Canterbury, New Zealand

[3]ISABELLE RIVALS & LeON PERSONNAZ, *Black-Box Modeling With State-Space Neural Networks,* ESPCI, Laboratoire d'Electronique, 10 rue Vauquelin, 75231 Paris Cedex 05, France.

[4]B.LENNOX, P. RUTHERFORD, G.A MONTAGUE AND HAUGHIN, *Case study investigating the application of neural networks for process modeling and condition monitoring,* Pergamon, PII: S0098 – 1354(98)00234-8,1998

[5]BARRY LENNOX, GARY A.MONTAGUE, ANDY M.FRITH, CHRIS GENT and VIC BEVAN, *Industrial Application of Neural Networks – An Investigation,* School of Engineering, University of Manchester, UK

[6]K. PIOTROWSKI, J. PIOTROWSKI, J. SCHLESINGER, *Modeling of complex liquid-vapour equilibria in the urea synthesis process with the use of artificial neural network,* Department of Chemical & Process Engineering, Silesian University of Technology, 44-100 Gliwice, Poland 2002

[7]S.ORESKI, J.ZUPAN, AND P.GLAVIC, *Neural Network Classification of Phase Equilibrium Methods,* Faculty of Chemistry and Chemical Engineering, University of Maribor, Solvenia, 2001

[8]W. KAMENSKI &E. TOMCZACK, *Evaluation of Drying and Degradation Kinetics Using Neurocomputing,* Brazilian Journal of Chemical Engineering, Print ISSN 0104-6632, 2000

[9]G. C. RODRIGUEZ[1], S. ESTRADA, M. A. GARCIA, AND M. A. SALGADO, *Dynamic modeling of drum drying using neural network-differential hybrid model,* Food Engineering Dept., Instituto Tecnológico de Veracruz, PO Box 1420, Veracruz , Ver., 91860, Mexico

[10]J.-S. ROGER JANG, *IEEE Neural Networks Council Standards Committee Working Group on Data Modeling Benchmarks, CS Dept.,* http://neural.cs.nthu.edu.tw/jang/benchmark/

[11]AL-DUWAISH, H., GHOUTI, L.,HALAWANI, T., AND MOHANDES,M.,*Use of artificial Neural Networks Process Analyzer: A Case Study,* ESANN'2002 Processing-European Symposium on Artificial Neural Networks,Bruges(Belgium),24-26 April 2002, ISBN 2-930307-02-1

[12]CARL G.LOONEY, *Pattern Recognition Using Neural Networks, Theories and Algorithms for Engineers and Scientists,* Oxford University Press, 1997

[13]SIMON HAYKIN, *Neural Networks, A comprehensive Foundation,* Prentice Hall, 1994

[14]NEURAL NETWORKS, StatSoft, Inc., 1984-2003, ttp://www.statsoftinc.com/textbook/stneunet.html

[15]WARREN SARLE, *Why use activation functions?* http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-10.html

[16]CHRIS WILLIAMS, *Data Preprocessing,* School of Informatics, University of Edinburgh

[17]DAVE ANDERSON AND GEORGE MCNEIL, *Artificial Neural Networks Technology,* Data &Analysis Center for Software, 775 Daedalian
http://www.dacs.dtic.mil/techs/neural/neural.title.html

[18]CHRISTOPHER M.BISHOP, *Neural Networks for Pattern Recognition,* Oxford University Press Inc, New York, 1996

[19]HENRIK JACOBSSON, NICKLAS BERGFELDT AND SIMON LUNDELL, *Matlab and Neural Network Toolbox Tutorial,* November 26, 2001

[20]J.M. SMITH, H.C. VAN NESS, M.M. ABBOTT, *Introduction to Chemical Engineering Thermodynamics,* McGraw-Hill Companies, Inc., Fifth Edition, 1996

[21]ECDB--*The Engineering Chemistry Data Base*, developed by the Laboratory of Computer Chemistry (LCC), the Institute of Chemical Metallurgy (ICM), http://mole.icm.ac.cn/#top

[22]ROBERT H. PERRY, DON W.GREEN, JAMES O. MALONEY, *Perry's Chemical Engineers' Handbook,* McGraw-Hill International Edition, Chemical Engineering series, Sixth edition, 1984

[23]AKTS AG. *Advanced kinetics and technology solutions,* TECHNO-Põle, 3960 Siders, Switzerland, http://www.akts.com/thermal-analysis

[24]Nurelegne Tefera, *Experimental data for temperature-time profile of adiabatic reaction, unpublished data*

[25]ANNE-JOHAN ANNEMA, *Feed-Forward Neural Networks, Vector Decomposition Analysis, Modeling and Analog Implementation,* Kluwer Academic Publications, 1995

[26]JEFFERY JOHNSON & PHILIP PICTON, *Designing intelligent machines,* Volume 2, Buttherworth – Heinemann Ltd., 1995

[27]JACK WINNICK, *Chemical Engineering Thermodynamics,* John Wiley & Sons, Inc., 1997

[28]B.G. KYLE, *Chemical and Process Thermodynamics,* Prentice Hall International Series in the Physical and Chemical Engineering Sciences, Second Edition, 1992

"The thesis is my original work, has not been presented for a degree in any other university and that all sources of material used for the thesis have been duly acknowledged"

Alemayehu Ambaw	_____	_____
Candidate	*Signature*	*Date*

Confirmed by:

Dr.Ing. Berehanu Assefa	_____	_____
Advisor	*Signature*	*Date*
Dr.Ing. Nurelegne Tefera	_____	_____
Advisor	*Signature*	*Date*