Real Time Temperature Prediction in a Data Center Environment using an Adaptive Algorithm

by

VISHOK AMAR KUMAR

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTERS IN MECHANICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2013

To my parents, Dr Amar Kumar J and Dr Nandini Chandappa and my beloved grand mother, Nagamani SV

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my advisor Dr.Alan Bowling , who has been my guide and mentor through this journey. Without his persistent help and guidance this dissertation would not have been possible. I would also like to thank my committee members Dr Agonafer and Mr Haji-Sheikh for whose work demonstrated to me that we can better the system prevalent in our environments by dedicated research towards finding better solutions. I would also like to thank Dr Manry, who gave me his valuable inputs with respect to the model of my system and helped me understand the fundamentals better. I would like to thank Swathi V Shenoy for her invaluable support and my lab mate Chinmay Date for working alongside me in developing this system and providing alternative solutions to benchmark the performance Finally, I would like to thank all my family and friends who have been a support and a source of motivation pushing me to excel in my undertaking.

November 22, 2013

ABSTRACT

Real Time Temperature Prediction in a Data Center Environment using an Adaptive Algorithm

Vishok Amar Kumar, M.S.

The University of Texas at Arlington, 2013

Co-Supervising Professors: Dr. Alan Bowling and Dr. Dereje Aganofer

Most organizations around the world rely on information systems to run their operations. Data centers are hence a crucial aspect of most organizational operations to ensure business continuity. Disruption in the working of a system can impair the business largely and hence optimizing the environment in which the servers and storage devises of a data center are housed becomes extremely challenging. Most industries design the data center infrastructure to handle peak load conditions thus ensuring a continual functionality of its hosted environment. However, the cost of maintaining such a system is very high. This gives rise to the need of having a robust system which could regulate the energy consumption and thereby reduce the cost of maintaining these environments ,at the same time guaranteeing optimal performance. Standardization of such a system will yield savings and help improve the design of the facility. In the traditional approach, a CFD model is used to model the dynamic and complex environment of a data center. This system however takes a considerably long time to converge to a steady state hence causing loss of productive time and resources. We propose to solve these issues by using both a feed forward network and

a dynamic recurrent artificial neural network which would mimic the functionality of the CFD , but at the same time guarantee a faster convergence rate and hence a better performance. Given a data center model with varying server heats and fan speeds ,our system is aimed to predict accurate temperature readings for the system ,hence ensuring controlled energy consumption in comparison to the traditional approaches. By using a recurrent system allowing time delays we incorporate the variables of a real time data center environment. Furthermore, the entire system is adaptive and hence the neural network would learn incrementally with every incoming data sample and use this error to better its performance. This guarantees an improvement of the system output and reduction in the prediction error with every incoming data sample as well as being scalable to model any room. This system is validated in against the results from the CFD and data from a static neural network for the same data center model. Various cases for different fan speeds and server heats have been tested and validated. This system can be incorporated to improve the control strategy for data centers.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

With the steep rise in web activity through social media, web based businesses, media applications and services, data centers have become a quintessential element in modern IT infrastructure. Practically every large IT organization has a data center in house or out sourced to vendors, especially services that require an always on capability. These factors have led to a tremendous growth in size, number and power consumptions of data centers. With substantially growing energy demands for high performance computing architecture and associated equipment, data centers have become a large consumer of electricity. They are also amongst the harshest on the environment with their enormous energy usage and large carbon footprint. (1-25) For instance, a recent study claims that web search uses half the energy equivalent to boiling a kettle of water. [1] The US Environmental Protection Agency, EPA, estimates that servers and data centers have consumed 1.5% of total US energy in 2006 with loads rivaling the output of 15 base power plants. Furthermore, the same report estimates the number of data centers to have doubled by 2011. [2] The EPA estimates that the energy consumption of data centers is doubling every 5 years amounting to 7.4billion by 2011. Furthermore, in addition to environmental degradation and soaring electricity bills, this increased power consumption may lead to system failures due to over heating or power capacity over load. Even when power distribution and cooling systems have reached peak capacity data centers continue to deploy high-density servers (e.g., blade servers) to cope with the ever-increasing load. Power consumption in a data server is a mix of many factors such as number of

options installed on the server, CPU activity, memory, disk drives and even the mix of instructions being executed [3]. Thus making data center research a prominent focus for optimized operation and design of a data center given a lot of importance.

As data centers are critical components of modern days IT infrastructure, it is imperative that the power demands, cooling and energy efficiency be managed effectively. State of the art data centers have the high degree of control at a room level such as liquid cooling and at a server level such as power distribution. Typical data centers have an average cooling capacity of 3kW per cabinet with a maximum of 10-15kW per cabinet whilst typical CRAC airflow supply to a cabinet is approximately $0.094 - 0.24 m^3/s$ (200-500 CFM). [4] In the not so distant future, to match the high computational performance demands, high performance chips with heat fluxes of $100 W/cm^2$ will increase heat loads at the server and facility level. Back in 2002, the heat load of a server compute cabinet was just 13kW but now has risen to over 28kW [5]. With increased heat loads, 30-50% of the energy consumption in a data center is targeted toward cooling [6] while the energy efficiency in most data centers are less than 50% [7]

Airflow in a data center is a complex process and is difficult to precisely map. Consequently the environment in a data center is a highly dynamic, intricate and complex thermal environment. Traditionally, Computational Fluid Dynamics have been used to model a data center and simulate its environment. Not only is this approach expensive in terms of software licenses but also in terms of computational power required to effectively run the software. CFD requires a trained user for accurate modeling. Typically, CFD takes considerable time to produce a steady state output. As stated earlier, the environment in a data center is highly volatile and dynamic and hence continuous simulation is a time consuming task. We propose to use a neural network pre-trained from CFD data, to simulate a data center's perfor-

mance. Further more, when deployed in a real world data center, the neural network will adaptively iterate itself based on real time data from the equipment in the room.

Neural networks are computational tools used for pattern recognition and machine learning. They are best used as a prediction tool. A NN will approximate a function between its inputs to produce a desired output. As shown in fig 1.1 it maps a series of inputs to its outputs through a system of interconnected weighted neurons and hidden layers containing activation functions. A Neural Network learns from past data and is trained to predict ouputs as close to the target output with a predefined error tolerance. Since the environment in a data center is highly non-linear and variable system, a NN is an ideal tool for modeling a data center's environment.

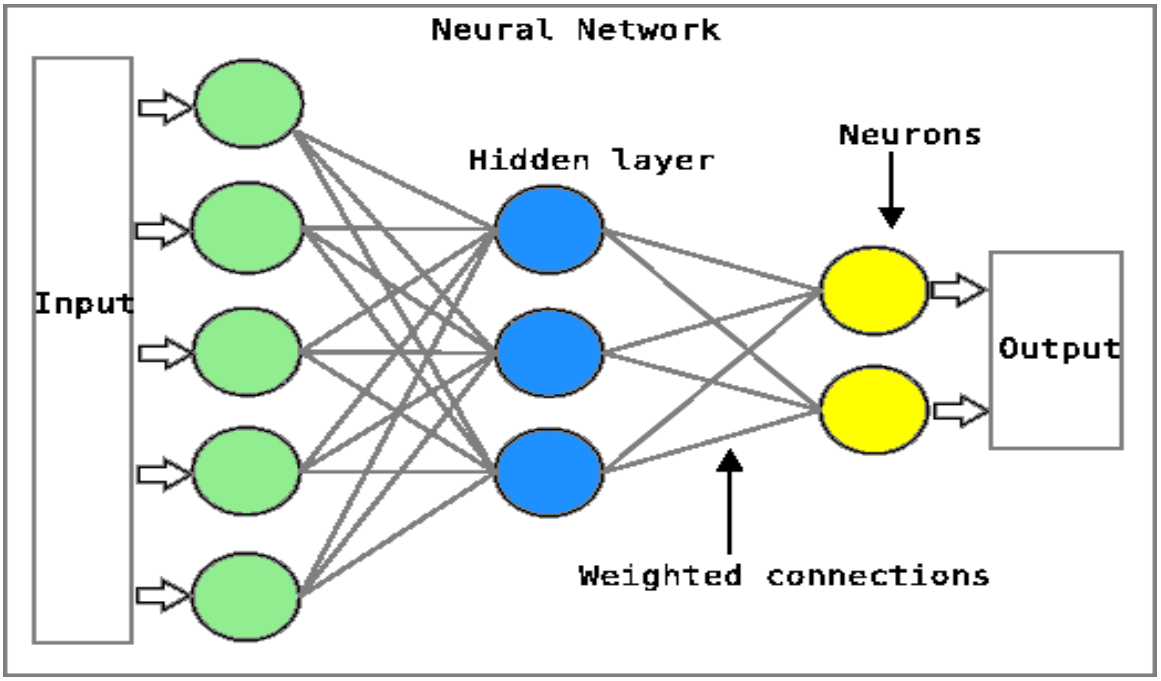

Figure 1.1. A Simple Neural Network.

CFD uses an algorithmic approach for computation, i.e it follows a set of instructions in order to solve a problem. Unless the specific steps that the CFD needs

to follow are known, it cannot solve the problem. That restricts the problem solving capability of a CFD model. Here a neural network is much more useful as they can do things that we do not know exactly know how to do. Neural networks learn by example. They process information in a similar way the human brain does. A large number of highly interconnected processing elements(neurones) work in parallel to compute the output. Another advantage is the high speed computational power of a NN compared to the time CFD takes to settle on a steady state output.

We propose to model and train two different NN architectures, a feed forward network and a recurrent neural network, to simulate the data center's environment and predict server air temperature accurately. A feed forward network is the most widely used NN with only forward connections as shown in fig 1.2. A RNN is a special type of NN that incorporates feedback from other neurons as well as its self. It is specially suited for dynamic systems as it has a time delay option in its structure to predict transient data which is especially effective in this application. Figure 1.3 shows a simple RNN.

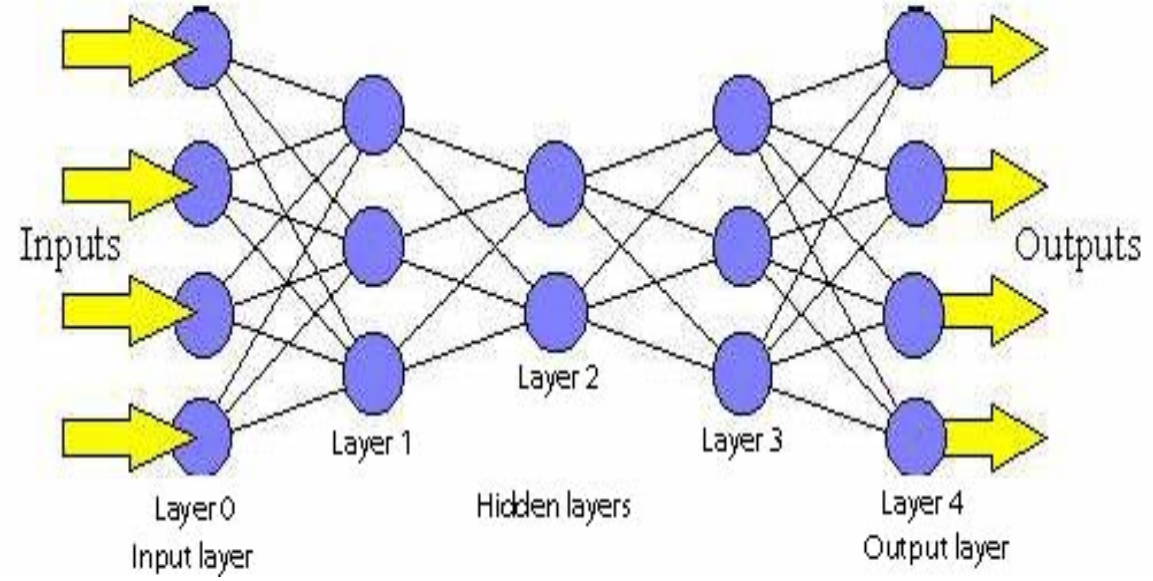

Figure 1.2. A Simple Feed Forward Network.

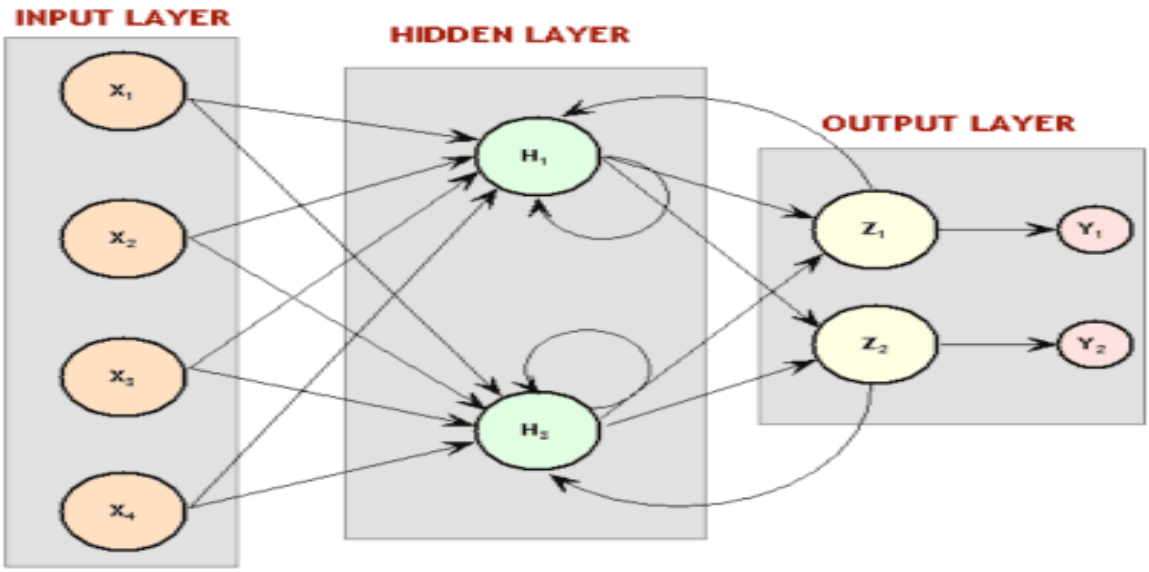

Figure 1.3. A Simple Recurrent Neural Network with Self Feedback.

When this system is deployed in a real time data center, we propose to incorporate an adaptive neural network that will receive real time data from the equipment in the center to dynamically adapt itself to predict accurate server inlet air temperatures. This network will actively update its weights corresponding to new targets it recieves from the equipment. This system is validated against CFD predictions and against predictions from a previously tested and validated batch trained NN for the same data center model.

CHAPTER 2

Background

2.1 ASHRAE Data Center Operation Guidelines

In recent years, there has been an exponential increase of activity in the IT domain. The stress in increased computer performance has unfortunately led to an undesirable side affect of high power. There are numerous factors affecting optimal data center operation ie server inlet temperature, air humidity, ambient temperature, server power utilization to name a few. The industry has recognized that with the increasing density with in the data center has a profound impact on the reliability and performance of the equipment it houses.

Initially, each commercial IT equipment manufacturer published their own operating recommendations for their equipment, typically operating in the range of 20-21 deg C. It was common belief that "colder is better". Also, many data centers use IT equipment from various vendors which proposes the problem of having to integrate varying manufacturer recommended operating environments. This created a quandary for customers to as to what environment to provide in their data processing room. A common ground was required for optimal and effective integration of these components in a data center.

To address this issue, ASHRAE (American Society of Heating and Air-Conditioning Engineers) proposed a set of guidelines to standardize the optimal operation of the equipment in a data center. The primary focus for the thermal management was the equipment's temperature and humidity requirements. ASHRAE developed four classes of data processing classes that encompassed most IT equipment. Classes 1 and

2 range from heavy duty air-conditioned server and storage environments while class 3 are for PCs, workstations and class 4 is equipment requiring virtually no environmental control. For each class environmental conditions such as allowable dry bulb temperature, relative humidty, maximum dew point, maximum elevation, maximum rate of change are specified. Non operating points of the same are also included. Also, psychometric chart of all environmental classes are provided. [8] [9]

2.2 CFD in Data Centers

Fluid flow is governed by three main aspects, (1) energy is conserved (2) mass is conserved and (3) Newton's second law (F=ma). These principles can be expressed in terms of partial differential equations. CFD in the most part is replacing these equations with numbers and advancing the system through time to obtain a description of the fluid flow of the system.

Computational Fluid Dynamics, more often than not abbreviated as CFD, is a branch of fluid dynamic is a fluid dynamics tool that uses FDM, FEM and FVM to solve, analyze and predict complex fluid flows. Computers are used to simulate the physics and predict time dependent results given initial set of boundary conditions 2.1. Experimental validation is carried out to validate initial CFD predictions.

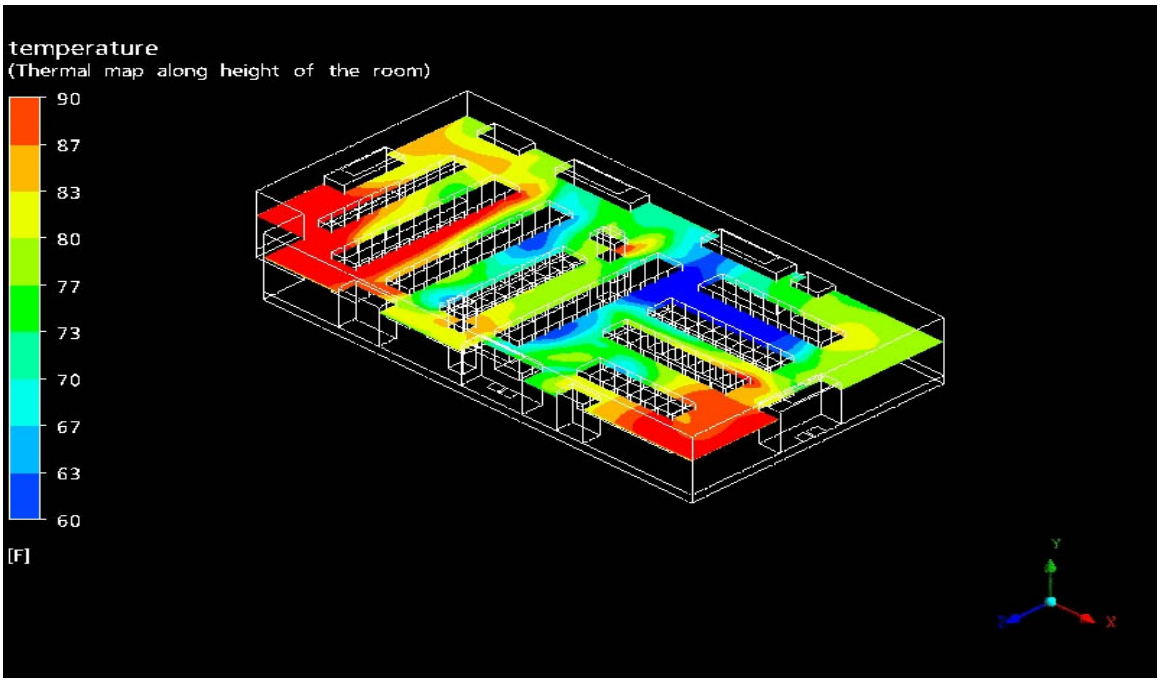

Figure 2.1. CFD Modeling.

It only seems logical to use such a robust and strong tool such as CFD to model data center environments. CFD has gained immense popularity to model cooling effectiveness within racks and aisles. It provides a 3-D analysis of how hot and cold air is moving through the data center and identifying regions that require additional cooling or areas that are excessively cooled. CFD gives the customer a prediction of how much cooling power is required for his application. An optimal layout of racks and containment can be derived from this analysis.

Some popular CFD software packages for data center environment analysis are:

- 6Sigma from Future Facilities
- Flovent from Flomerics Group
- TileFlow from Innovative Research Inc. of Minnesota.
- CoolSim software from ANSYS

2.3 Room Layout

Here we describe the reference data center configuration that has been modeled. The current configuration hosts 40 racks arranged in a hot aisle/cold aisle layout. The room is symmetrical across the middle section as shown in fig

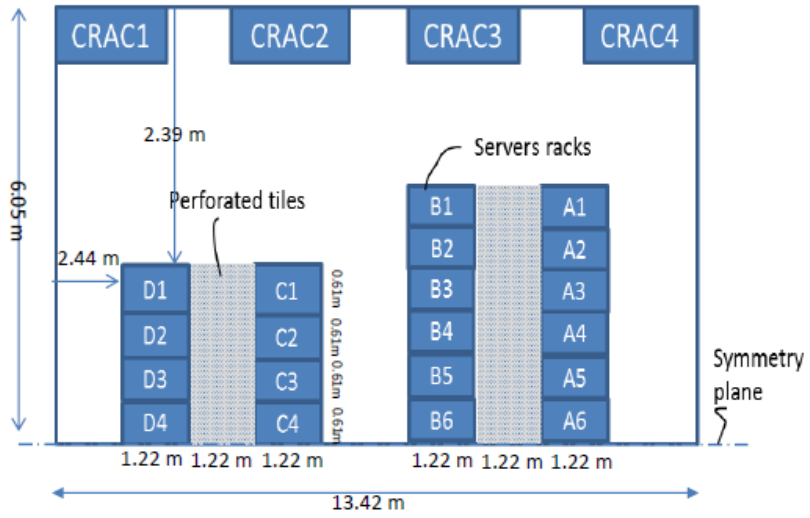

Figure 2.2. Top view of the data center layout.

The room is 13.42 m in length, 6.05 m in width and 3.66 m in height. Each rack is 0.61 m wide, 1.22 m wide and 2.0 m high. Each rack has 6 temperature data points where the temperatures are measured for analysis. These temperature data points are 0.333 m apart on a rack.

2.4 Past Work

A batch trained NN for the same data center model has been developed, tested and validated. The NN is modeled on a feed forward architecture. The model is trained purely on CFD data and tested for predictions within the data set. This model has an accuracy of about 98/

2.5 Why MATLAB?

MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming as defined by MathWorks,Inc. It is an advanced and robust software package designed for engineering and scientific computation. Matlab does a good job of integrating graphical and numeric display of results. Integrating computation, visualization and programming in a easy to use environment making MATLAB a robust and favourite package in the engineering and scientific community.

Typical uses included but are not limited to:

- Modeling and Simulation
- Mathematics and Computation
- Data analysis and visualization
- Algorithm development
- Engineering and Scientific Graphics

CHAPTER 3

Neural Networks

3.0.1 An Introduction

A neural network (NN) is a information processing system composed of single elements operating in parallel. These networks are a derivation of the biological nervous system. Just as in nature, it is comprised of a highly interconnected network of processing elements called neurons. These neurons work in unison to solve a specific problem. Their function is highly dependent by the connections between these elements. In the nervous system, learning is achieved through specific adjustments to the synaptic connections between neurons. Similarly in neural networks, we train a NN to perform a particular function by adjusting the values of weights between the elements. [10] [11]

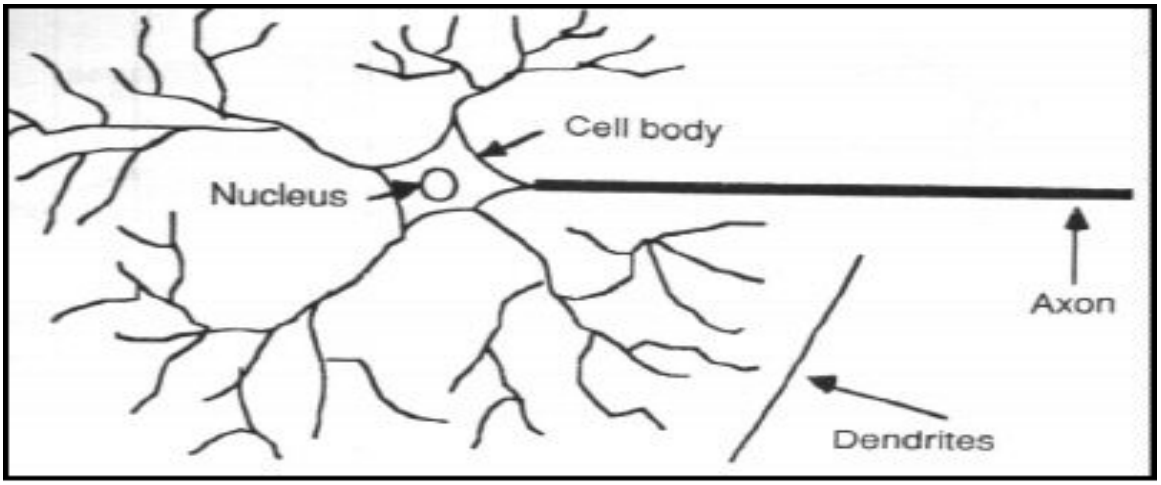

Figure 3.1. Components of a Biological Neuron.

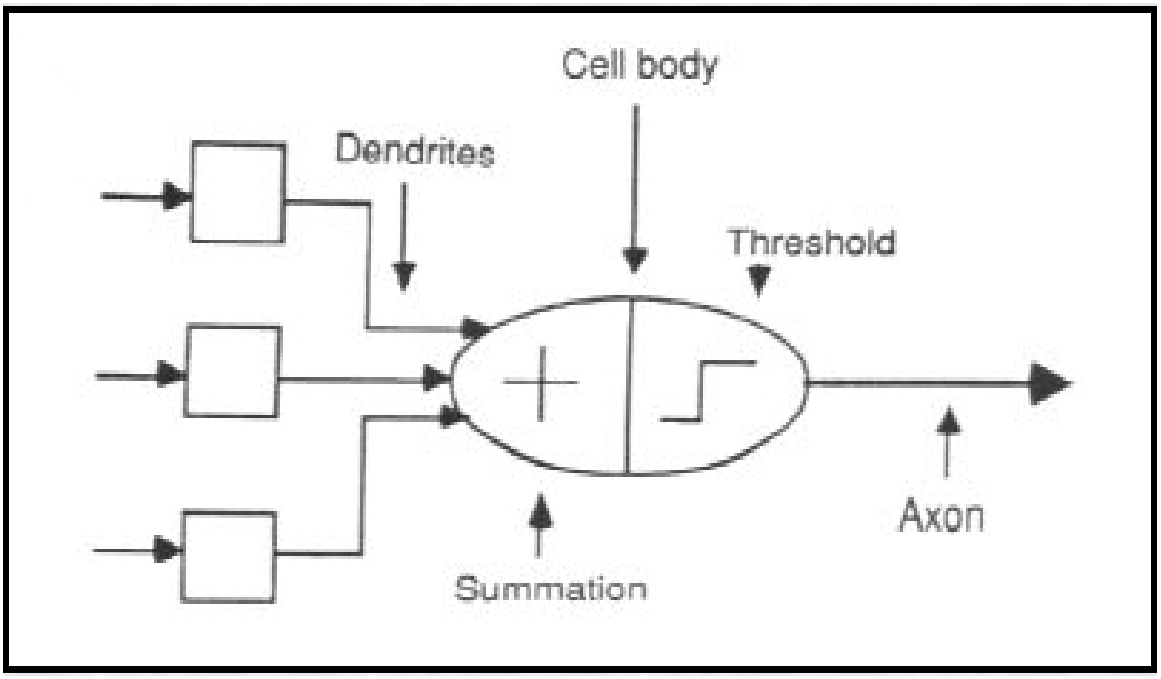

Figure 3.2. Components of a Neuron Model.

Neural networks are conventionally used where traditional computing techniques fail. They can not do everything but they can some things that otherwise would be very difficult. They can form a model and output predictions based on their training data or even their input data in the case of adaptive networks. NNs have been used in numerous fields such as aerospace, automotive, banking, electronics, defense, medical, financial, speech recognition, and telecommunications. [12] A few specific applications include high performance auto pilot systems, credit card evaluations and chip failures analysis.

3.0.2 Neural Network Design

The work flow for a conventional NN has been classified into seven primary steps. These steps cover the development of the NN to the execution and validation of the NN. The steps are:

1. Collect data
2. Create the network
3. Configure the network
4. Initialize the weights and biases
5. Train the network
6. Validate the network
7. Use the network

3.0.3 Network Architecture

The spatial arrangement and connection pattern of neurons within and in-between layers is called the architecture of the net. Weights, neurons, interconnects, inputs and outputs together form a network. Networks can as simple as a single layer or more complicated multiple layers. The number of layers in a network is the number of layers of weighted interconnected links between slabs of neurons. **??** If two layers with weighted interconnects are present, then there will be a hidden layer in between them.

NN come in various shapes and sizes: Feed forward, recurrent networks, fully interconnected networks etc. Some common networks are shown in fig 3.3 :

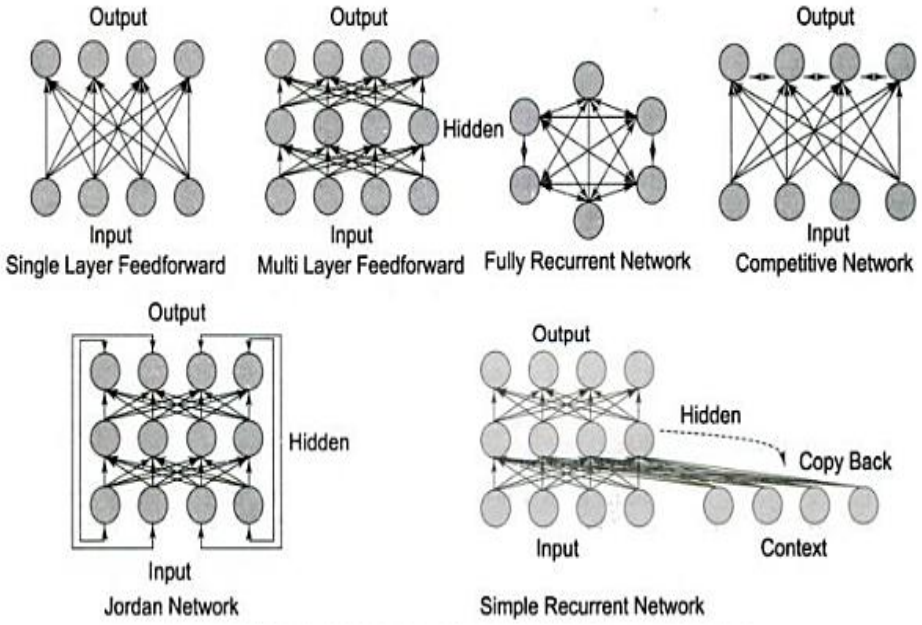

Figure 3.3. A fully connected recurrent neural network.

3.0.3.1 Feed Forward Net

Feed forward networks a single layer with the inputs directly connected to the outputs or multiple layer of input weights with hidden layers in between them. Hidden layers and their associated weights are used to create a representation of the input patterns.

1. Single Layer Network: This is the simplest type of network. It consists of a single layer if input nodes that feed via weighted interconnects directly to a single layer of output nodes. In a single layer net, the weights from one output does not affect the weights of another unit. They are capable of learning only linearly separable problems.

2. Multi Layer Network: Here there is one or more layers on nodes between the input and output layer called the hidden layer. These networks can solve more complicated problems.

3.0.3.2 Competitive Net

This network is similar to the feed forward network except that there are usually negative connections between the output nodes. These connection are usually restricted to units that are close to one another. Due to these connections the output nodes tend to compete to represent the current input pattern. Using a apt learning algorithm, the system can reorganize itself topologically. [13]

3.0.3.3 Recurrent Net

Recurrent networks are time dependent networks. They have the ability to connect all neurons to each other and even themselves of need be. They process sequential information ie the state of the network depends on the previous time step.

3.0.4 A Neuron Model

The basic building block of a neural network is a neuron. A neuron consists of a set of inputs, weights in the input layer and output layer, a bias, an activation function and the output. A host of interconnected neurons together constitute a neural network.

3.0.4.1 Neuron Architecture

A scalar input p is multiplied by a weight w to form a weighted input wp. b is a scalar bias that is much like a weight and generally has a value of one. It is simply added to the input or to the transfer function f shifting it to the left by the amount

b. The arithmetic sum of the weighted input wp and the bias b form the input to the transfer function and is denoted by n. The transfer function f is generally a sigmoid function or step function. It takes the input n and gives an output a. The value of wp and b can be adjusted to accordingly to so that the NN exhibits a desired behavior.

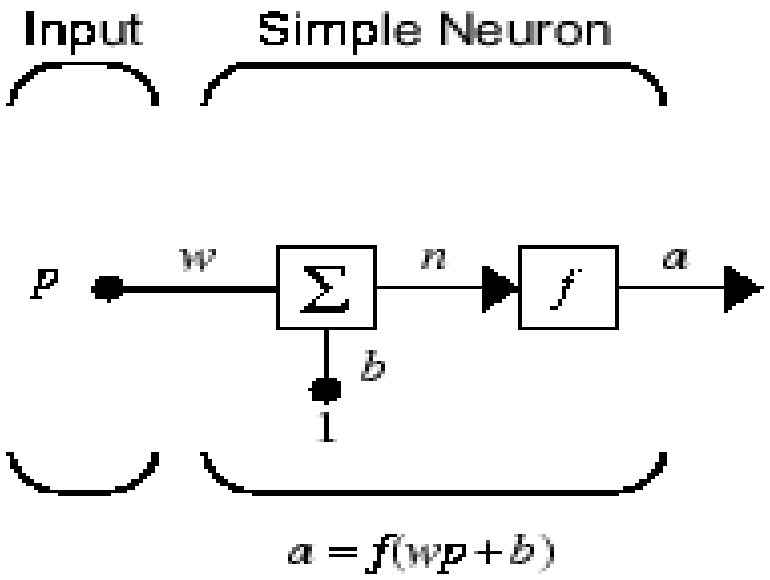

Figure 3.4. Model Neuron with a Scalar Input.

NN can have vector inputs as shown in fig 3.4. They are prominent in adaptive neural networks. The sum Wp is the dot product of the weighted values matrix W and the input vector p. The neuron has a bias b which is summed to give the net input n.

3.0.4.2 Transfer Functions

Transfer functions also known as activation functions contain adaptive parameters that are optimized. Larger networks with simple neurons may have the same power as smaller networks with more complex neurons. /ref transfer functions. A

transfer function is best chosen by by trial and error to see which gives the a more consistent and desirable output. A recent list of known and new transfer functions have been published recently /ref 12 from same paper. There are three main types of transfer functions, a hard limit transfer function, linear transfer function and sigmoid transfer function.

1. Hard Limit Transfer Function Here the output of the neuron is limited to 0 if the net input n in less than 1 or is 1 if the input n is greater than or equal to 0.

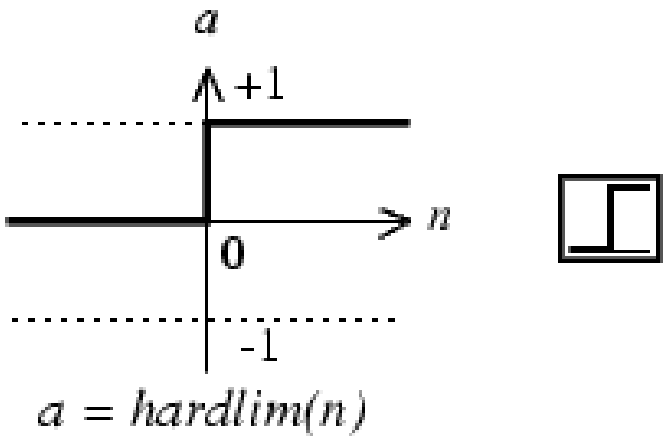

Figure 3.5. Hard Limit Transfer Functions.

2. Linear Transfer Function Here the output is conditioned according to a linear equation

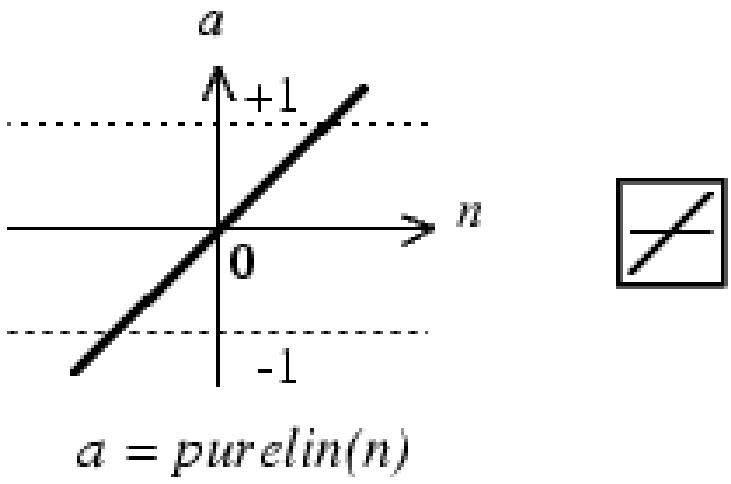

Figure 3.6. Linear Transfer Functions.

3. Log Sigmoid Transfer Function It is a differentiable transfer function and hence is desirable in back propagation techniques. Here the takes a the input n which could vary between plus to minus infinity and limits the output between a range of 0 and 1.

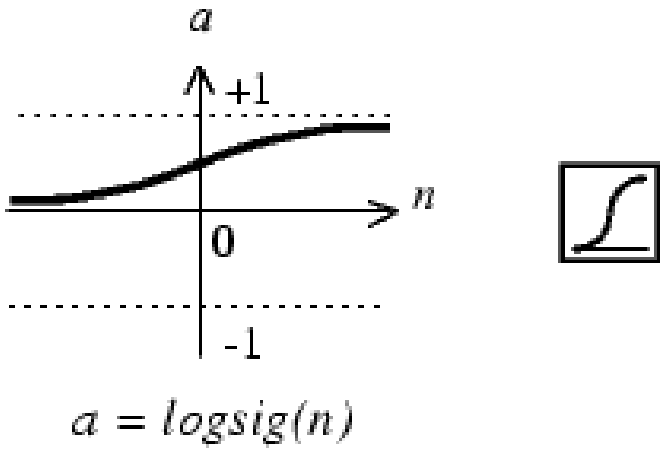

Figure 3.7. Log Sigmoid Transfer Functions.

3.0.4.3 Bias

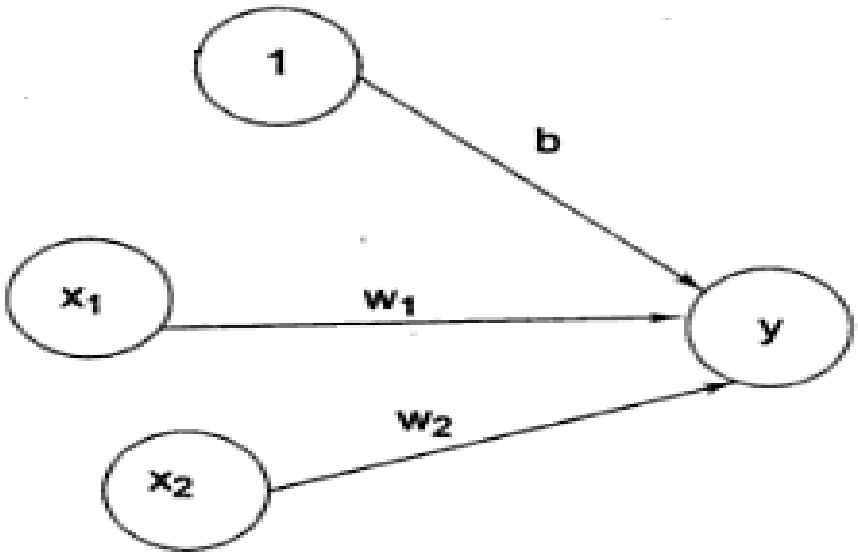

Figure 3.8. Neuron with Bias.

A bias acts just as weight on connection whos activation is always equal to one. They improve the performance of the NN. Increasing the bias increases the net output to the unit. The bias is initially initialized to zero or any other value depending on the network. The net input from the network is calculated as:

$Net = b + \sum x_i w_i$

Where, Net : input b : bias x_i : input from neuron i w_i : weight of neuron i to the output neuron

3.0.5 Error Estimation

The error calculations used to train a NN is very important. There have been numerous error calculation methods experimented with. So for any set of input of

values and weights there will be a corresponding error associate with it. This error is dependent how well the NN has been trained ie which structure has been used, which learning algorithm is used. The Delta Rule applies an error function called gardient descent. The negative of the error associated to a certain weight is proportional to the change applied to that weight.

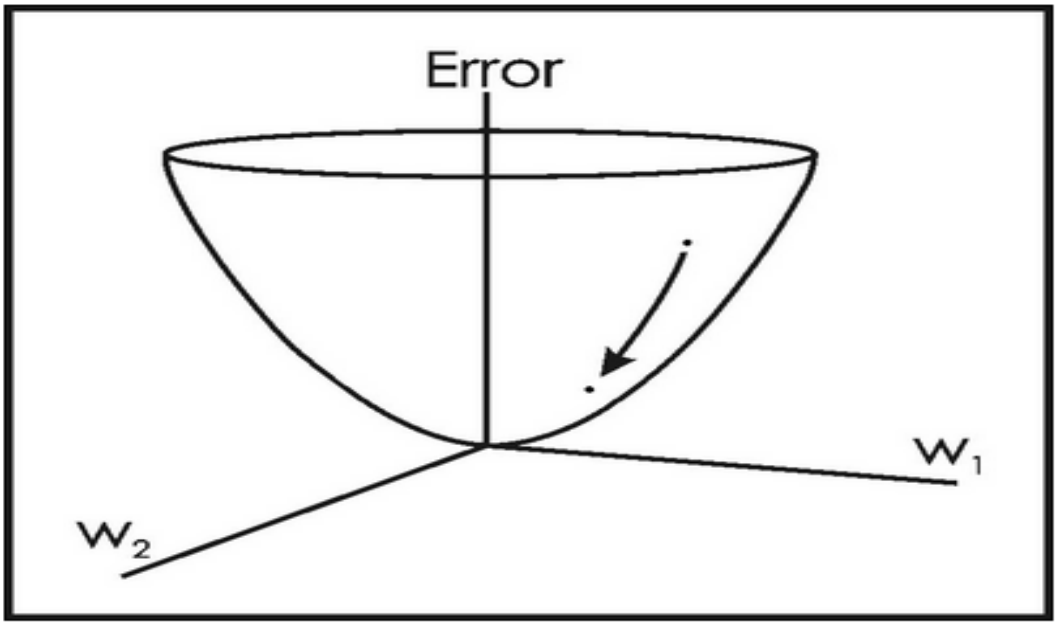

Figure 3.9. Schematic for Error Function Containing Two Weights (w1 and w2).

The error function most commonly used is the mean square error (MSE). This is the squared difference between what the NN predicts for each iteration and its corresponding target value. The equation is:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{C}(T_{ij} - t_{ij})^2 \tag{3.1}$$

Where: N : is the total number of training cases C : is the number of outputs T_{ij} : $target value for i^{th} node and j^{th} layer$ t_{ij} : $predicted value for i^{th} node and j^{th} layer$

3.1 Feed Forward Networks

Feed forward neural networks (FFNNs) are the most well known and widely used class of neural networks. The popularity of feed forward networks is derived from the fact that they have been applied successfully to a wide range of information processing tasks. A feed forward neural network is an artificial neural network where connections between the units do not form a directed cycle as in fig **??**. Feed forward networks have a characteristic layered architecture, with each layer comprising one or more simple processing units called artificial neurons. All of these networks have input layer and a output layer. It is the simplest type of NN where the information moves from only in one direction ie from its input layer through the hidden layer (if any) to the output layer. There are no cycles or loops in the network.

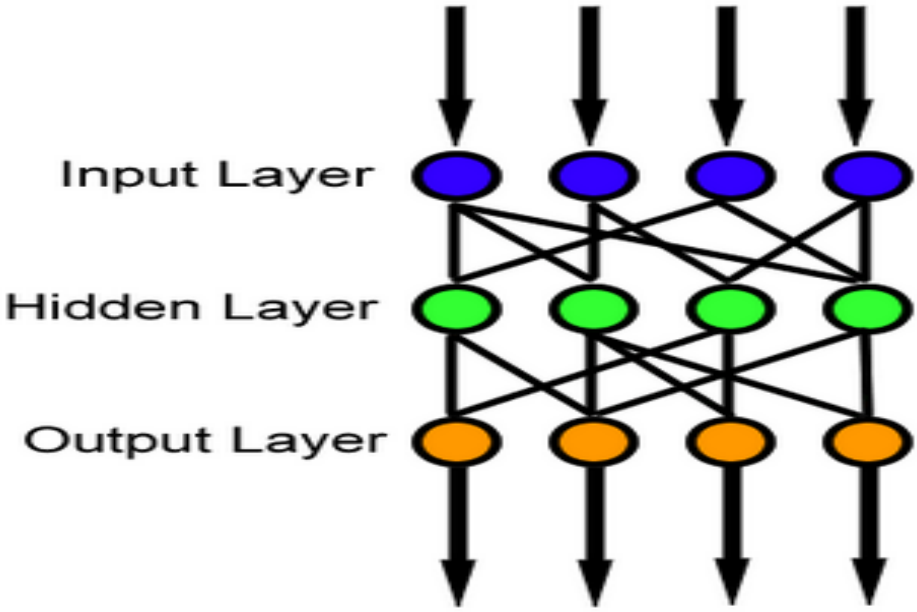

Figure 3.10. Feed forward Neural Network.

3.1.1 Single Layer Feed Forward Networks

A neural network is a network of neurons organized with multiple layers. In its basic form, it consists of just one input layer that connects to an output layer of neurons, but not vice versa. Figure 3.11 shows a single layer FFNN with four nodes in the input layer and output layer. This is called a single layer network with the single layer referring to the output layer of nodes. We do not count the input layer of nodes as no computation is performed here.

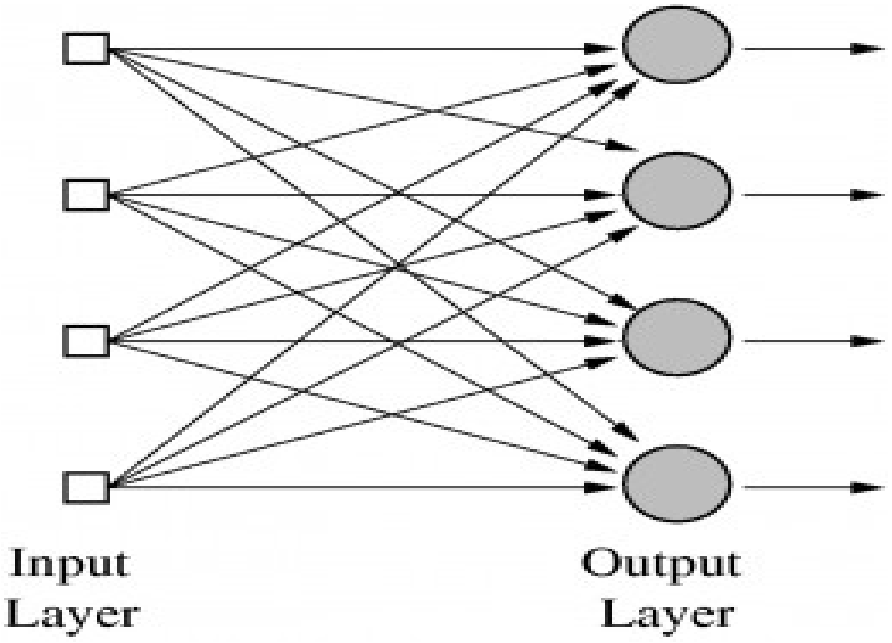

Figure 3.11. Single Layer FFNN.

3.1.2 Multilayer Feed Forward Networks

The second class of feed forward neural networks are multilayer feed forward networks. They have one or more hidden layers that are called hidden neurons. Their function is to intervene between the input layer and output layer for increased

performance. With hidden layers, the NN is able to extract higher order statistics. This ability to extract higher order statistics is valuable when the input layer is large.

The primary sources in the input layer of the NN give the respective elements the input vector which act as the input signals applied to the computational nodes in the second layer (the first hidden layer). The output of the second layer is used as input to the third and so and so forth. The signal set from the final layer are considered the response of the network. Figure 3.12 shows the architecture of a a multi layer feed forward network with a single hidden layer. The network is termed as a 4-5-1 network with 4 inputs, 5 hidden neurons and 1 output neurons. This network is said to be fully connected in the sense that every node in the each layer is connected to every node in the adjacent layer.

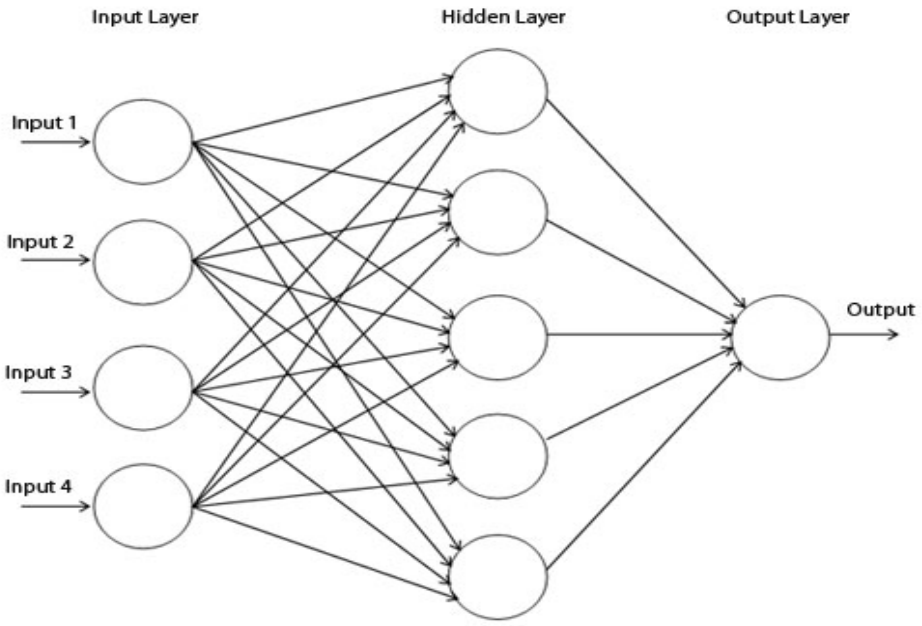

Figure 3.12. Muliti Layer FFNN.

3.2 Recurrent Neural Networks

3.2.1 An Introduction

Recurrent Neural Networks (RNNs) are designed to learn sequential or time dependent results. They utilize temporal data processing whose computational units use activations based on the activation history of the network. RNNs distinguishing characteristic is their ability to map a series of input sequences distributed across time to their outputs sequences. In other words, for a same input parameters the RNN could predict different outputs dependent on the time sequence in reference. RNNs offer a framework suitable for resulting network outputs values in training **??** [14]

Using a RNN in server temperature prediction is typical use of RNNs. A data center's environment is heavily dynamic. Typically loads on the servers fluctuate considerably through out working day and would cause an corresponding environment change. The CRAC would have to react to these changes to maintain the the servers in their optimal operating environments. An RNN works great with dynamic systems and has the ability to work with time dependent data. For a certain set of inputs from the data center, it could predict the room's environment as a function of time. This is helpful as we can predict transient states as well as steady state states of the servers.

3.2.2 Architecture

RNNs can be fully connected (3.13) or partially connected networks, including feedforward networks. Fully connected networks do not have distinct input layers to nodes but have inputs from other nodes. They could even have feedback to themselves [15].

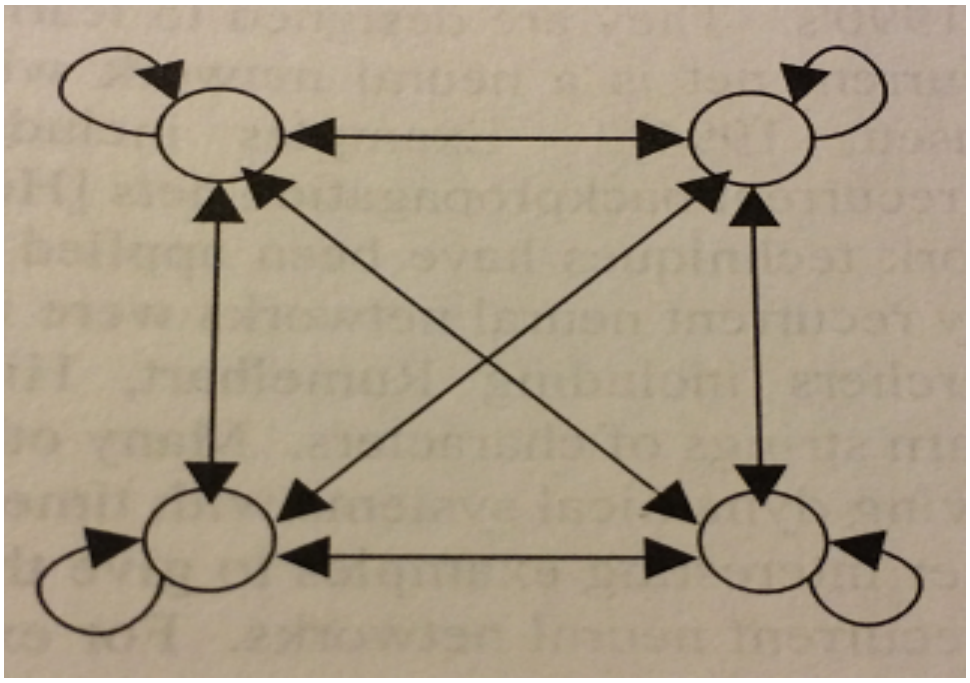

Figure 3.13. A fully connected recurrent neural network.

A simple RNN have a feed forward like structure but receive feedback from other nodes and sequential context from other nodes. 3.14 shows a context layer C1 and C2 where the weights to these layers can be processed through back propagation. This layer retains information between observations [16]. As new inputs are fed into the RNN, the previous contents of the hidden layer are passed into the context layer. These are then fed back into the hidden layer at the following time step. The context layer receives time delayed feedback from the following layer of units.

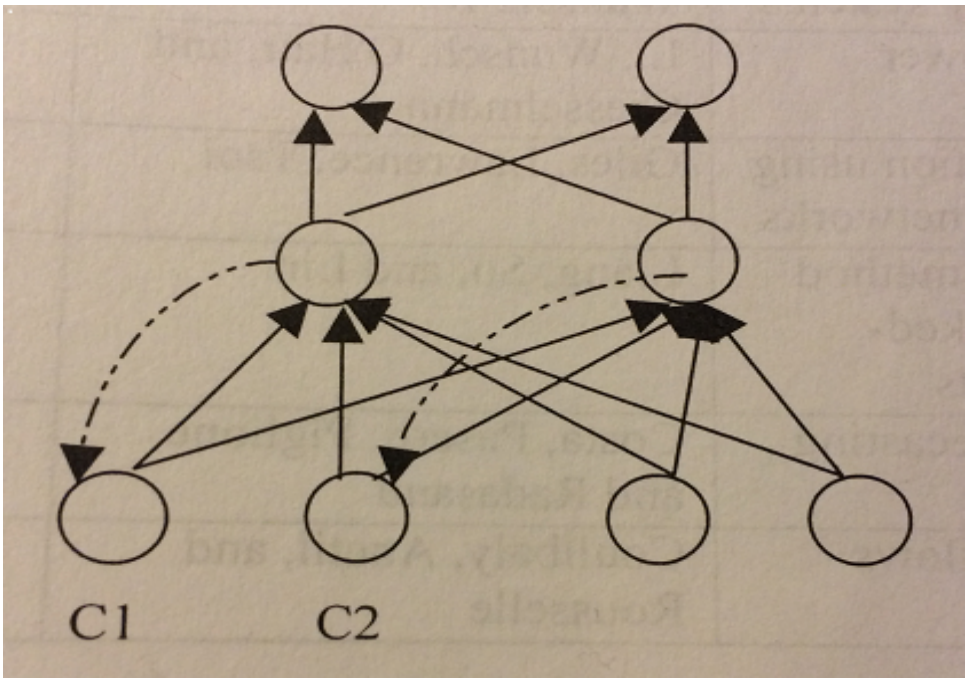

Figure 3.14. A simple recurrent neural network.

3.2.3 Learning in Recurrent Neural Networks

Neural networks are attractive in their applications because of their inert ability to learn and continue learning. Different learning algorithms are used for specific applications [17]. Exploiting the RNNs computational capabilities is a challange its own. To take full advantage of the application of the a RNN one approach is use a feed forward format integrated. Researchers have developed a variety of methods of gradient methods of which back propagation is most effective. Back propagation through time was proposed back in 1990s. [18]. Understanding how RNNs process information is imperative when deciding which training algorithm to use.

CHAPTER 4

Methodology

4.1 Incremental Learning

4.1.1 An Introduction

In numerous applications learning algorithms must act in dynamic environments where data will be continuously generated. New information arrives in separate batches over time. [19] Traditional batch learned algorithms re learn the concept from scratch and can not be used. A more suitable approach is incremental learning

4.1.2 Description of the Method

Here the network at in its current state $Net(t_k)$ can be updated when ever a new data set T_{k+1} is available allowing the resulting network $Net(t_{k+1})$ to include both the new data T_{k+1} and data from its previous data set. This network is known to be adaptively learning.

4.1.2.1 Initial Training of the Network

The matrix of weights W_1 corresponding to the weights on the connections between the output nodes and the preceding hidden layer nodes. The *sum of squares* error is now minimized. The sum of squares error is the sum of the squared differences between each observation and its group's mean. This error is a quadratic function of

weights and its minimum can be found by solving a set of linear equations in terms of a pseudo-inverse matrix [20] [21]

$$E = \frac{1}{2}\sum_{i=1}^{q} 1 \sum_{n=1}^{N} 1[o_i(x_n - T_i(x_n]^2 \tag{4.1}$$

Where, $T_i(x_n) - i^{th}$ target value , $o_i x_n - i^{th}$ output of the network

4.1.2.2 Incremental Backpropagation Neural Network

A key feature in designing a incremental learning network is learning new knowledge with out forgetting the old. Incremental back propagation network is one such network. It combines structural adaption learning rules, bounded weight modification and initial knowledge to curb the learning process [22] [23] Simple back propagation is not incremental in its nature. If it trained on input set A and then retrained on input set B, the knowledge of data set A maybe lost. The back propagation algorithm must be modified so that the when the network is retrained whilst keeping the knowledge of the previous training state. The idea is to minimize the network output error with respect to old input sets subject to the approximation of the network output to the desired output of the new input sets. Since the output of a NN is determined by how its neurons are connected and the weights on those neurons. Adaptation is based on two aspects, weight adaption and structural adaption. Using bounded weight modification and structural adaption an incremental back propagation network is developed.

4.1.2.3 Bounded weight update

A bound on the weight modification is implemented so that the previous knowledge of the network is retained while allowing margin for updating the weights. More

the uncertainty in the network, larger the weight bounds should be. As the learning proceeds, the weights should begin to and eventually converge.

Two different learning strategies can be implemented for error minimization, fast learning and slow learning. In fast learning, the weights are adjusted on every instance as long as they are with in their bounds and slow learning is where one weight change is done per epoch. Fast learning is not as effective as slow learning.

The network learns by back propagation with in its weight bounded limits. The network may not shift to its steepest descent because the weights will be pruned. To avoid this, a scaling factor **s** is introduced so that all scaled weights adjustment is within bounds. The learning rule is:

$$\Delta W_{ij}(k) = s(k)\eta\delta_j(k)O_i(k)$$

where ΔW_{ij} is the weight change from unit i to j, η is the learning rate, δ_j is the error gradient at unit j and O_i is the activation at i, all for the k^{th} iteration The weight adjustment for at an instance is $\mathbf{W(p)}$ such that:

$$|\sum_{k=1}^{n} \Delta W_{ij}(k)| \leq B_p$$

where n is the number of iterations run. Therefore the scaling factor **s** for the n^th iteration is introduced.

The weight bound is based on prior knowledge or determined by the learning curve. The bound should be set that the learning curve is smooth. It is concluded that bounded weight modification is needed for incremental learning since simple back propagation with such a constraint can not learn incrementally. Ref paper.

4.1.2.4 Structural Adaptation

The simple back propagation algorithm depicts how to change the weights on a neuron but not its structure. One way is do deactivate the neurons upon adaption where the adaption is below a certain threshold. How ever this approach fails where a

new neuron is added. In incremental learning, the weights prior to learning represent the learning so far. If the NN can not be adapted by weight adaption then structural adaption becomes necessary.

If the NN can not learn a new instance by weight adaption then a new hidden unit is added which encodes the input-output characteristics of the instance. The input units present in the instance are connected to the added hidden units with an initial weight *1/p* where p is the total number of attributes. That hidden unit is then connected to the output with a small weight. If the output weight is based on a single instance is reduced to a small value then the unit is removed. Although if the NN can only be increasing in size, it will reach a limit where its performance is compromised due to the spatial or temporal limitations.

CHAPTER 5

Training of the Neural Network

For the development of a NN model, the following problems need to addressed [24]

- How many hidden neurons in the hidden layer should be used?
- Which training algorithm should be used?
- What neural network architecture should be used?

All these factors can affect the error on the nodes to which the output is connected. The minimal error reflects better stability of the NN and a higher error reflects instability of the NN. The error before adaptive learning begins will tend be higher and will minimize as the networks adapts to the desired targets.

Two different neural network architectures have been trained and tested with the adaptive algorithm namely a feed forward structure and recurrent neural network structure. The performance of these architectures are compared on similar grounds.

5.0.3 Training the FFNN

The network is trained from CFD simulated data entered in a vector format. The data consists of several trials of uniform server power loading, non uniform server power loading, CRAC fan speeds, tile perforation ratios and target temperatures of the inlet air. The entire data set ranges from 5kW to 35kW of dissipated server heats, 60%, 80%, 100%, 120% CRAC fan speeds all at 50% tile perforation ratio.

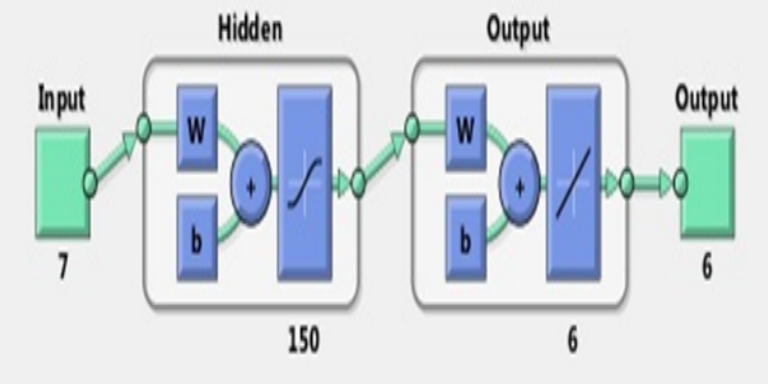

Figure 5.1. Matlab Model Representation for the Feed Forward Architecture.

The network is then tested for predictions outside its training set but with in the data range for robustness.

Then the network is trained to a new plant model feeding it inputs on a case by case basis. The adaptive NN learns the new room with every new test case fed to it. The NN is tested for extreme loading predictions where the batch trained NN failed. The batch trained NN is modeled as the plant to feed multiple inputs. 80 such cases were generated and fed to the NN as data.

For a training case of 5kW at 80% CRAC fan speed is now run through the adaptive NN for 5 passes. Figure 5.2 shows the iterative predictions for 5 passes of the adaptive NN for all rack 1. It can clearly be seen how the NN is learning with every iteration whilst adjusting its weights accordingly. Each iteration is run in a matter of seconds and can be adapted to a real time setting. The average error per iteration for 120 temperature data points is shown in table 5.1.

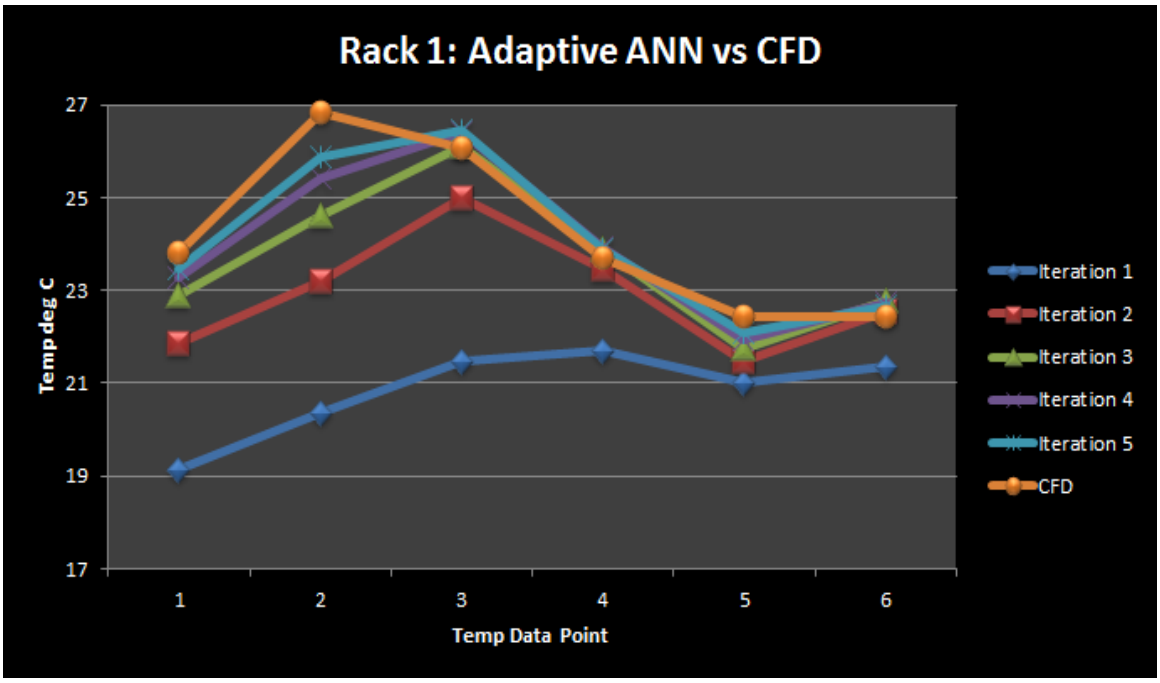

Figure 5.2. Iterative Temperature Predictions for Rack 1 Compared to CFD Predictions.

Table 5.1. Mean Error Reduction per Iteration

Iteration 1	2.072395833
Iteration 2	0.761734167
Iteration 3	0.444356667
Iteration 4	0.276921667
Iteration 5	0.1930675

Table 5.2 and Table 5.3 shows the update of weights in the NN from iteration 1 to iteration 2 for rack 1. These are small increments or decrements to each weights on the neurons between the hidden layer and output layer.

Table 5.2. Weight Matrix for Iteration 1

Hidden Neuron	1	2	3	4	5	6	7	8	9	10
T1	0.0583	-1.1851	-0.0002	0.8003	-0.0067	0.3714	0.0269	0.0647	0.5767	0.4427
T2	-0.5420	-0.2829	0.1620	0.9094	0.2275	0.2300	0.4766	-0.5446	0.8325	0.2810
T3	0.0284	-0.9014	-0.4167	0.9487	-0.0928	-0.0621	0.2176	-0.1468	0.0961	0.2072
T4	0.0807	0.2861	-0.4533	0.9652	-0.3214	-0.3650	-0.0570	0.0823	0.2207	0.3615
T5	0.1627	0.5373	-0.4076	0.9497	-0.3566	-0.5005	-0.2396	0.2374	0.2477	-0.1100
T6	-0.2039	0.3135	-0.3649	1.0141	-0.3320	-0.6380	0.0514	-0.1296	0.4782	0.3013

Table 5.3. Weight Matrix for Iteration 2

Hidden Neuron	1	2	3	4	5	6	7	8	9	10
T1	0.0549	-1.1814	-0.0013	0.8022	-0.0082	0.3723	0.0275	0.0662	0.5774	0.4466
T2	-0.5477	-0.2769	0.1602	0.9125	0.2251	0.2315	0.4774	-0.5422	0.8336	0.2875
T3	0.0317	-0.9049	-0.4156	0.9468	-0.0914	-0.0630	0.2171	-0.1482	0.0954	0.2034
T4	0.0820	0.2848	-0.4530	0.9645	-0.3209	-0.3653	-0.0571	0.0818	0.2205	0.3601
T5	0.1609	0.5392	-0.4082	0.9507	-0.3574	-0.5001	-0.2393	0.2382	0.2481	-0.1079
T6	-0.2006	0.3100	-0.3639	1.0123	-0.3306	-0.6389	0.0509	-0.1310	0.4700	0.2976

5.0.4 Training the RNN

The neural network was designed as a recurrent neural network with one hidden layer that has a recurrent connection with a tap delay associated to it. This allows the network to have an infinite dynamic response to the time series input data. The RNN is given seven inputs (server load x 4, fan speed, spatial position of rack x2) per iteration. After numerous trials, the hidden layer size is set at ten. The RNN returns six temperature data points per iteration as shown in fig 5.3. The RNN is trained via adaptive incremental learning. The time delay is set to zero as this network is not trained for transient data but could be used for the same.

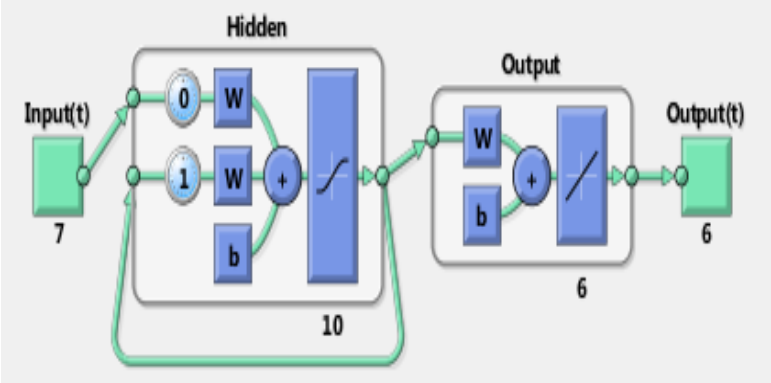

Figure 5.3. Recurrent Neural Network.

After many trials, it is found that 10 hidden neurons yield improved accuracy and stability of the network. This size is used for all test cases that have been performed.

The network is trained from CFD simulated data entered in a vector format. The data consists of several trials of uniform server power loading, non uniform server power loading, CRAC fan speeds, tile perforation ratios and target temperatures of

the inlet air. The entire data set ranges from 5kW to 35kW of dissipated server heats, 60%, 80%, 100%, 120% CRAC fan speeds all at 50% tile perforation ratio.

The network has been trained for 1000epochs and is continuously trained for better results. Figure 5.4 shows the training performance of the RNN. It achieves a lowest MSE of 1.6702 at its last epoch.

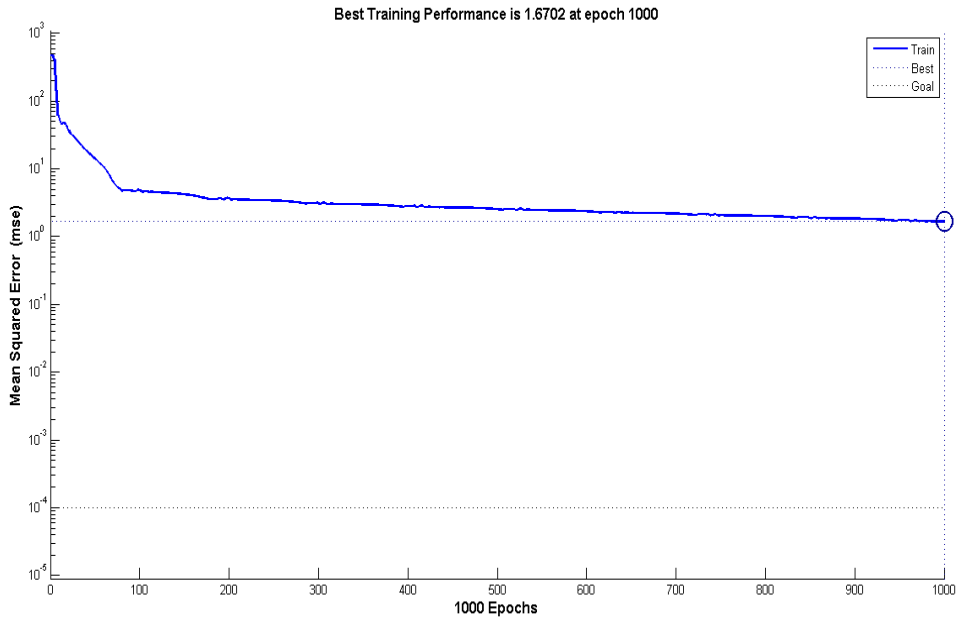

Figure 5.4. Recurrent Neural Network.

The learning rate of the RNN is at its highest at around epoch 70 and cycles there after. The learning rate gradually reduces a shown in fig 5.5. This is also reflected in fig 5.4 where the MSE reduces drastically till epoch 70 and there after almost linearly decreases but at a more relaxed rate.

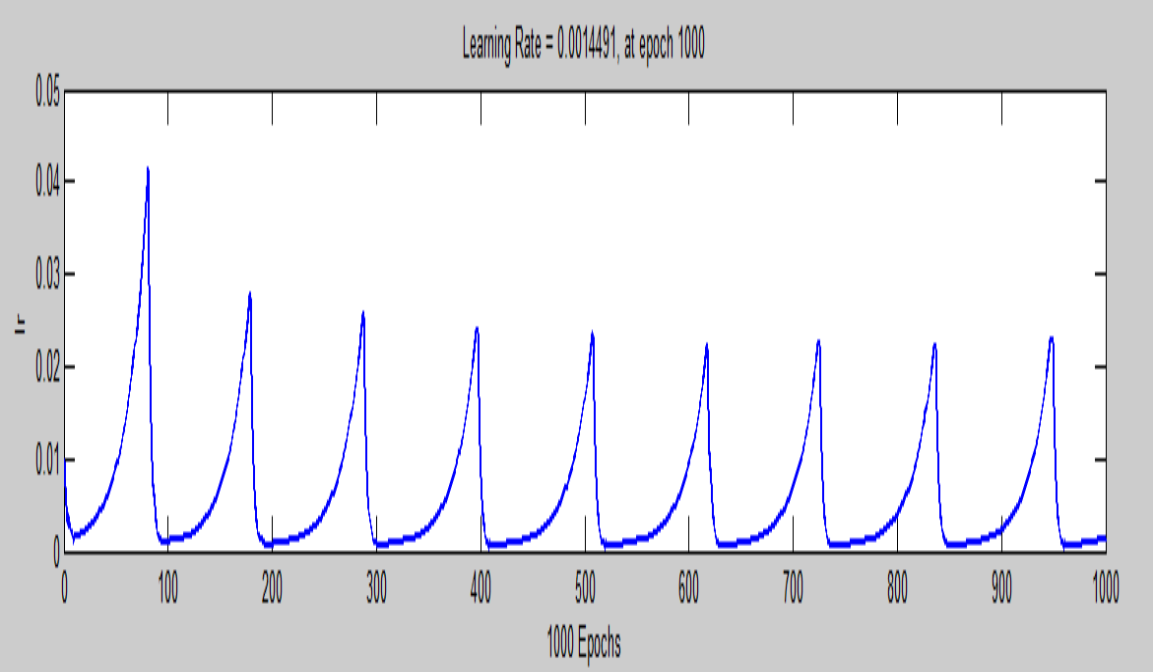

Figure 5.5. Learning rate of the Recurrent Neural Network.

Figure 5.7 shows that maximum erros lay between $+0.3934 Celcius and -0.3805$ Celcius for the data set. Considerable errors do lay in the $\pm$ 2.0 elcius range as well. The regression plot shows that this network is accurate for temperatures where enough data was available. Temperature points in the extremes (>30deg C) show some inaccuracy. See fig 5.6. To mitigate these errors is where the adaptive phase of the NN comes into play.

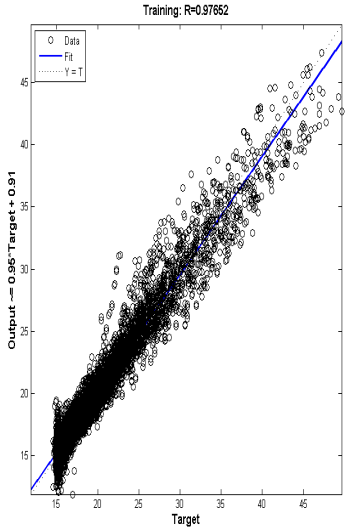

Figure 5.6. Regression.

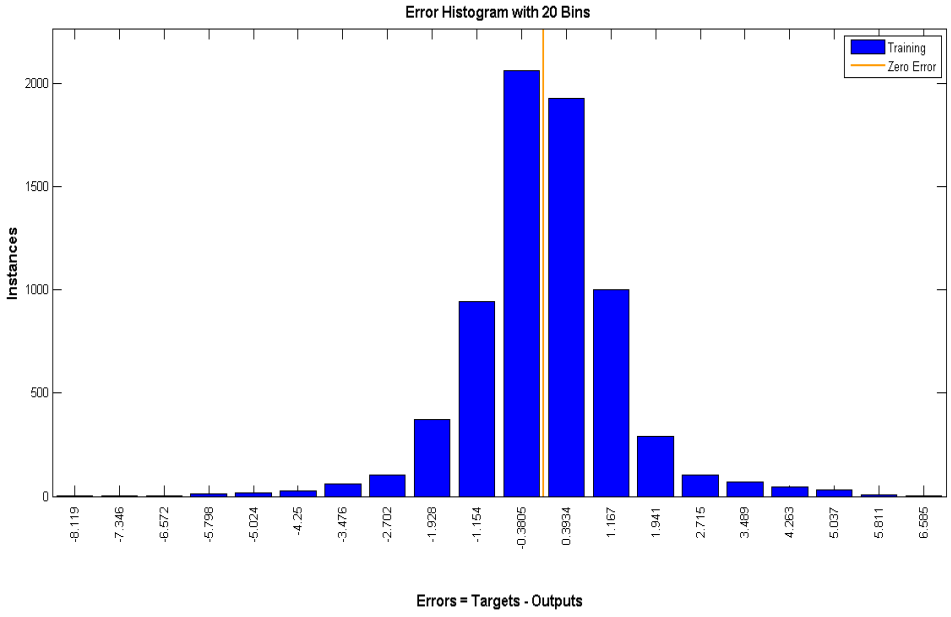

Figure 5.7. Recurrent Neural Network.

Figure 5.8 shows the stack up of 5 iterative RNN ANN predictions and the CFD prediction for the same case. It can be seen the shift of the successive iterations toward the target.

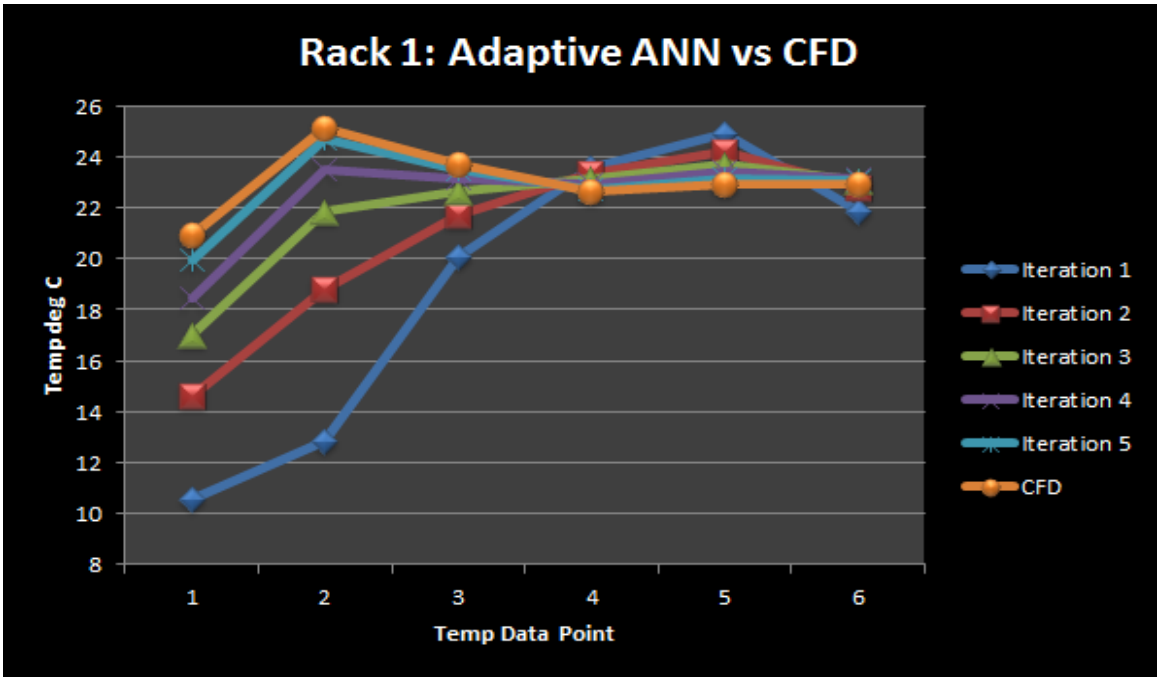

Figure 5.8. Iterative Temperature Predictions for Rack 1 Compared to CFD Predictions.

CHAPTER 6

Test Cases and Verification

6.1 Feed Forward Architecture

This architecture is tested for predictions with in its data set to test the performance of the network and for predictions outside its data set to test for the robustness of the network. The network is further tested for scalability and testing outside its data for predictions off training via the batch trained black box model.

6.1.1 Testing for CFD data

Testing done in two parts, for predictions from data set and predictions for data not in data set but with in range of data set.

6.1.1.1 Performance Testing

Three distinct test cases were chosen for testing the performance, 5kW per rack at 60% CRAC fan speed, 10kW per rack at 100% CRAC fan speed, 20kW per rack at 100% CRAC fan speed. These predictions were compared to CFD predictions for the same setting. The adaptive NN predictions are shown in figs 6.1 6.2 and 6.3. The average error percentage for all predictions is below 1% for test cases. The maximum error is around 1 degree C for all cases.

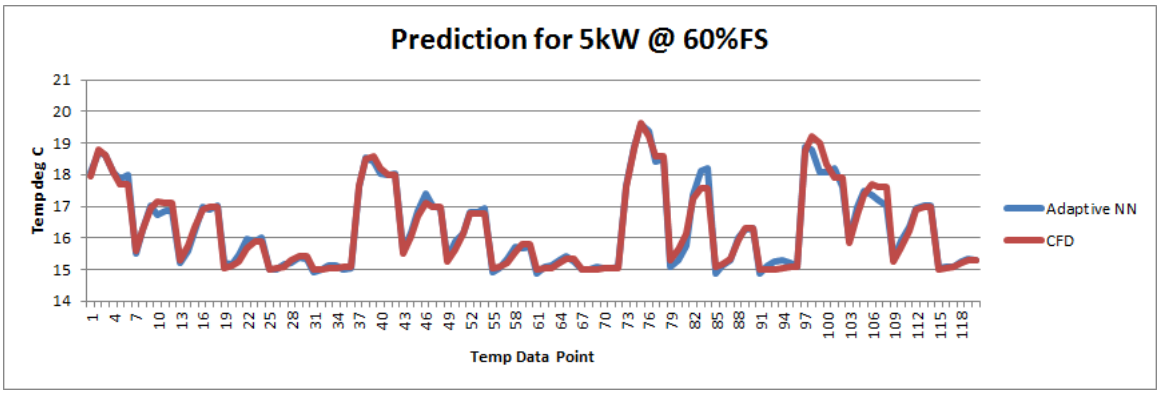

Figure 6.1. Prediction for 5kW at 60% CRAC fan speed.

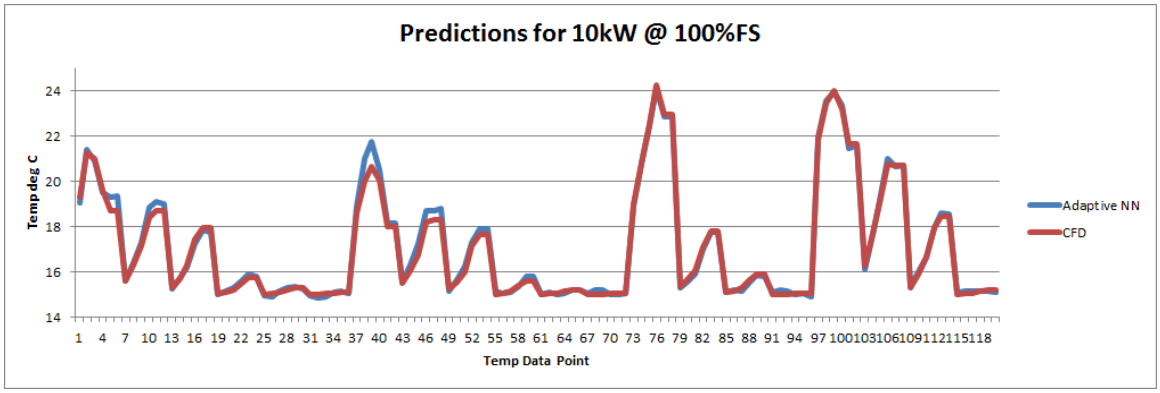

Figure 6.2. Prediction for 10kW at 100% CRAC fan speed.

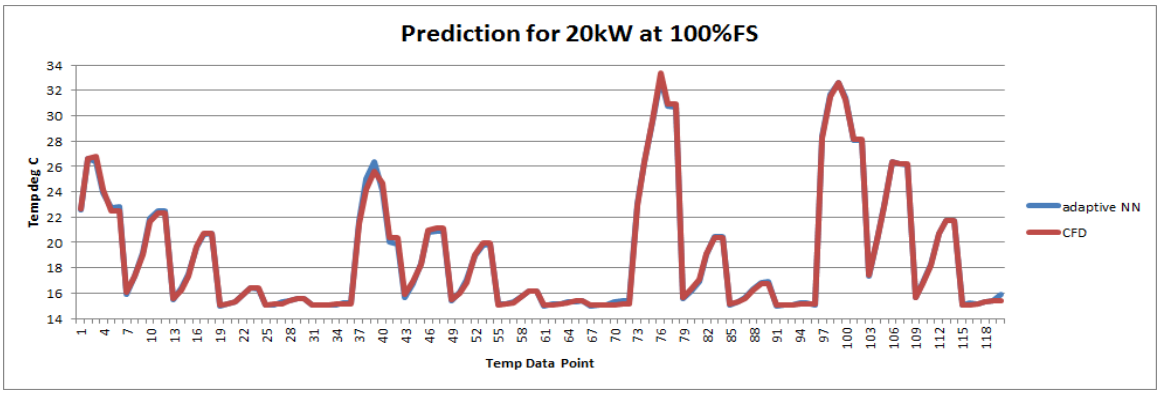

Figure 6.3. Prediction for 20kW at 100% CRAC fan speed.

The errors for these predictions are shown in figs 6.4, **??** and 6.6. The error plot 6.7 shows the averaged error percentage for the test case for 120 temperature data points. The error percentage is averaged at 0.74% with a maximum of 5.37% and a minimum at 0.007%.

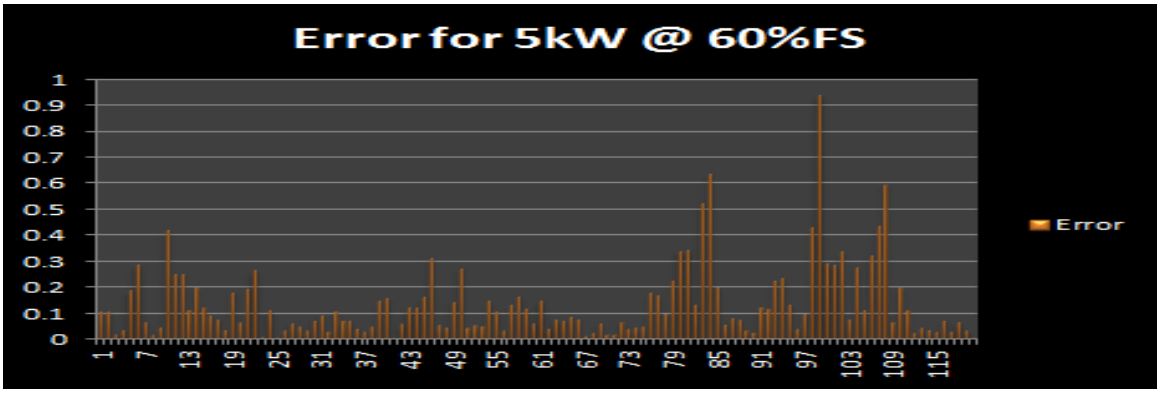

Figure 6.4. Errors for 5kW at 60% CRAC fan speed.

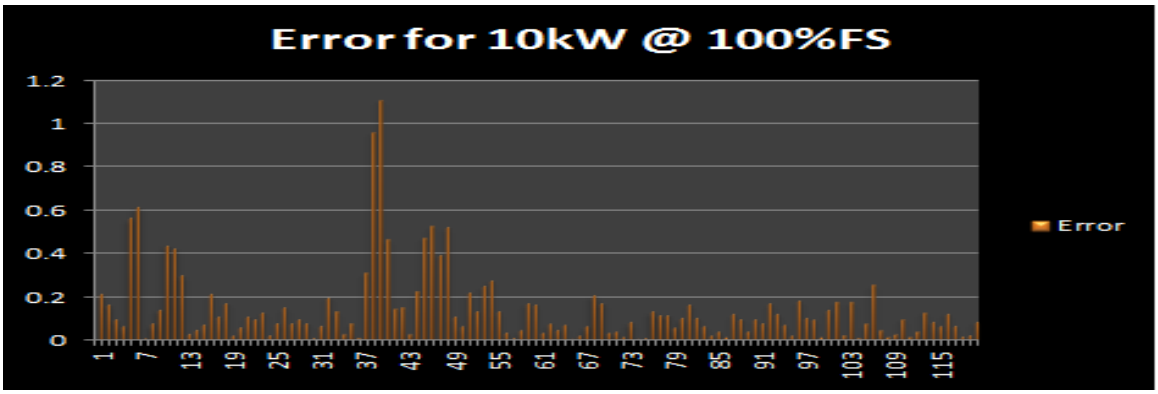

Figure 6.5. Errors for 10kW at 100% CRAC fan speed.

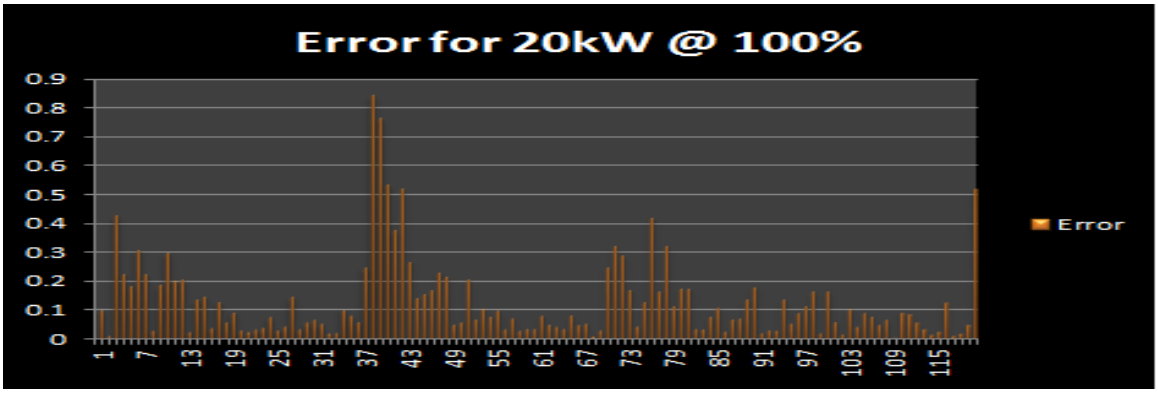

Figure 6.6. Errors for 20kW at 100% CRAC fan speed.

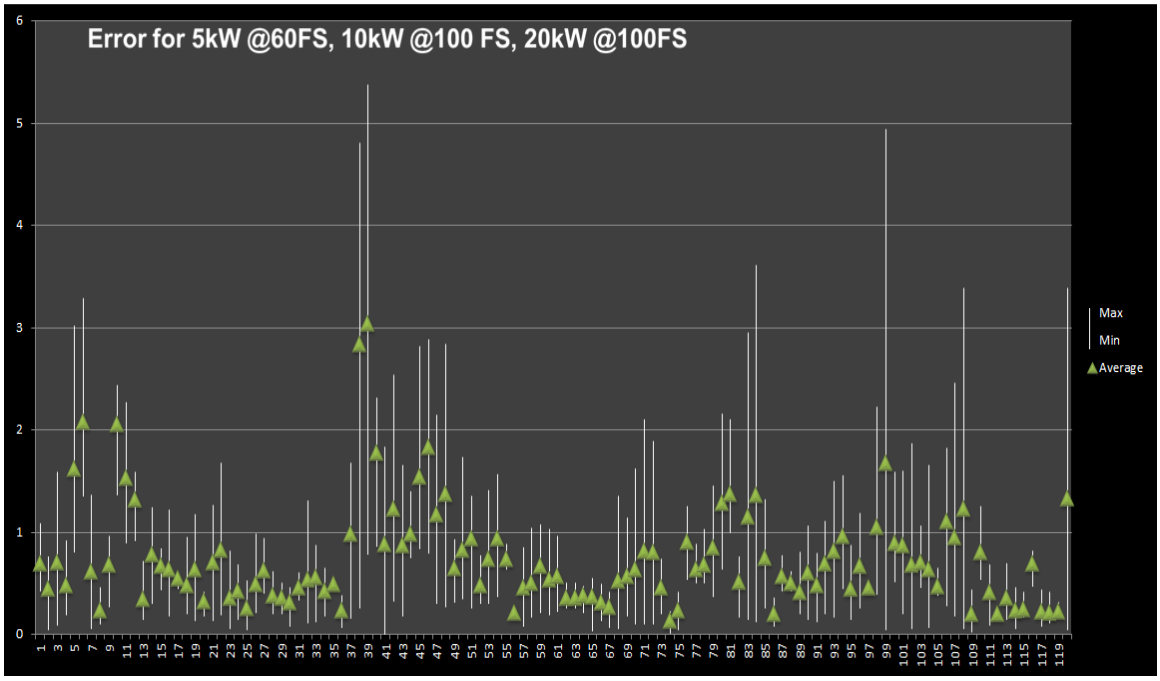

Figure 6.7. Error Percentage Chart.

6.1.1.2 Testing for Robustness

Three test cases are chosen for intermittent testing with in the range of the data set that are not included in the training data set. The cases are 12.5kW per rack at 60% CRAC fan speed, 17.5kW per rack at 80% CRAC fan speed and 22.5kW per rack at 100% CRAC fan speed. The NN predictions were compared to CFD predictions for the same setting. The adaptive NN predictions are chosen in fig 6.8, **??** and **??**. The average error percentage for all predictions is still below 1%. The maximum error is below 1 degree Celcius.

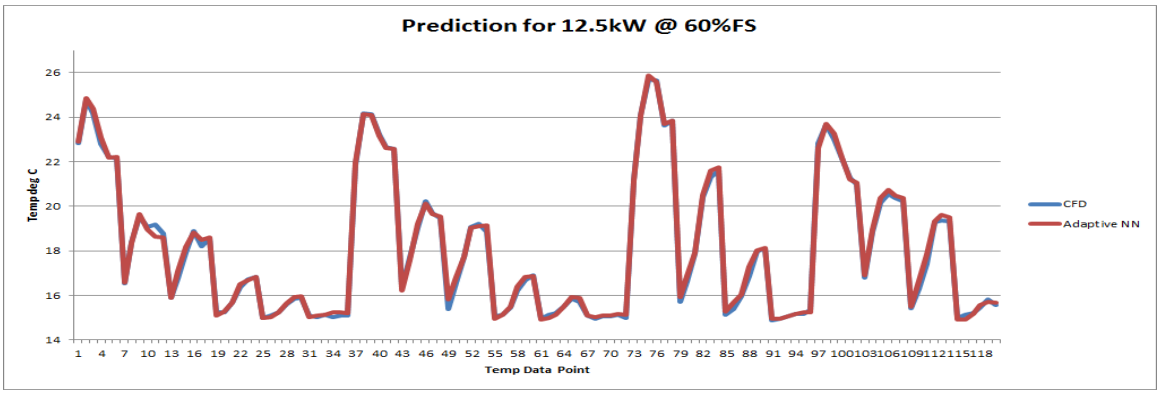

Figure 6.8. Prediction for 12.5kW at 60% CRAC fan speed.

The errors for these predictions are shown in figs 6.9, 6.10 and 6.11. The error plot 6.12 shows the averaged error percentage for the test case for 120 temperature data points. The error percentage is averaged at 0.72% with a maximum of 3.74% and a minimum at 0.0006%

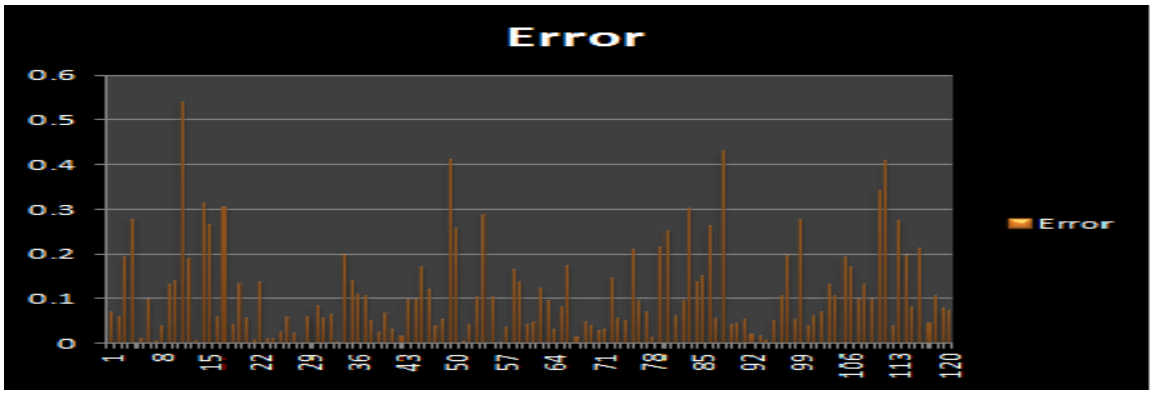

Figure 6.9. Errors for 12.5kW at 60% CRAC fan speed.

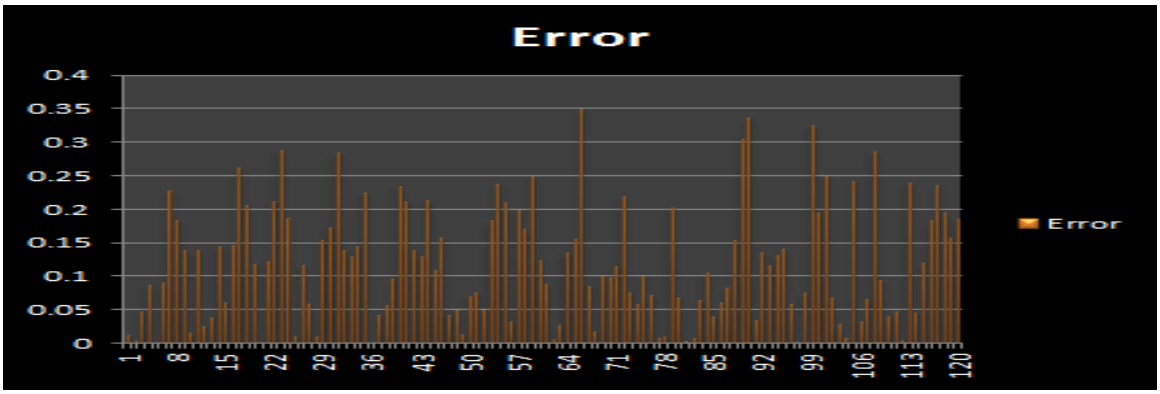

Figure 6.10. Errors for 17.5kW at 80% CRAC fan speed.

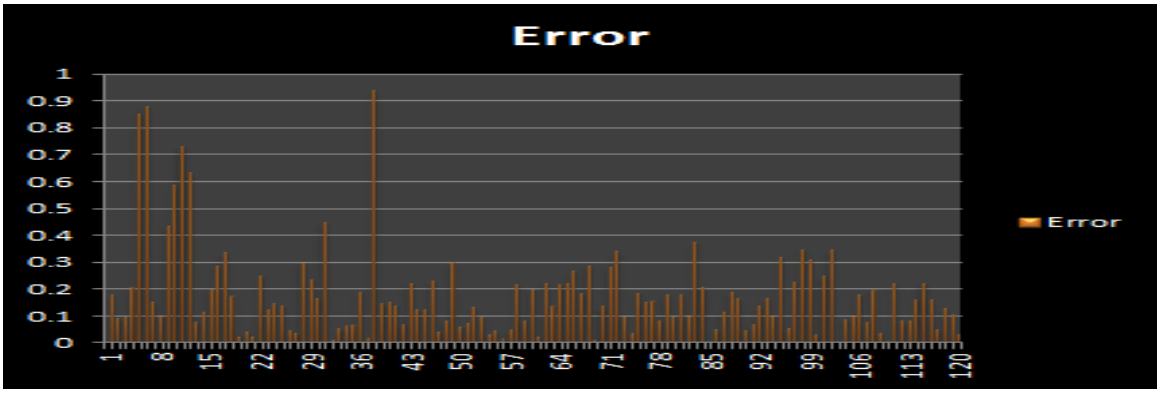

Figure 6.11. Errors for 22.5kW at 100% CRAC fan speed.

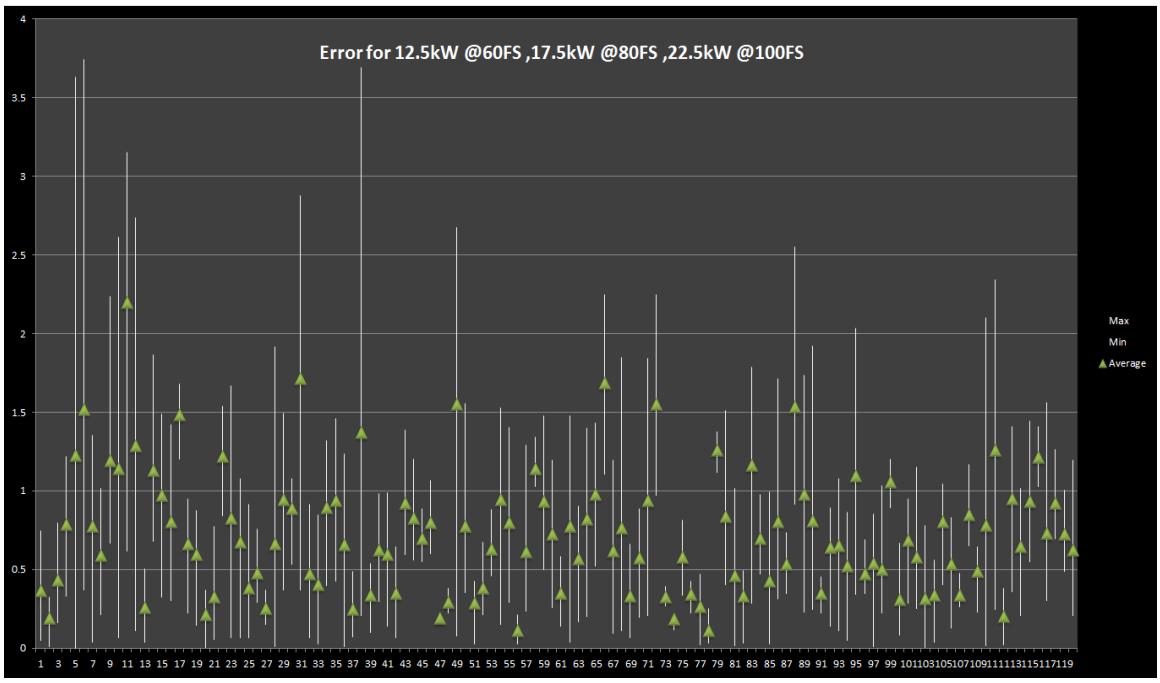

Figure 6.12. Error Percentage Chart.

6.1.2 Scalability

Here the scalability of the network is tested. A batch trained NN modeled around the same data center is used to simulate a plant. The plant can generate infinite predictions for infinite inputs. These inputs are fed to a nascent NN on a case by case basis. On feeding the NN enough data, it will be sufficiently well trained. Fifteen test cases are chosen to test the performance of the network. The test cases are combination of extreme loading conditions and intermittent fan speeds. The test cases chosen are shown below in table XXX. The performance of the network is highly accurate. The average error percentage 0.19%. The maximum error is only 0.54 degree Celsius while the minimum is at 0.003 degree Celsius. Figure 6.13 shows the average error percentage for all 15 trials.

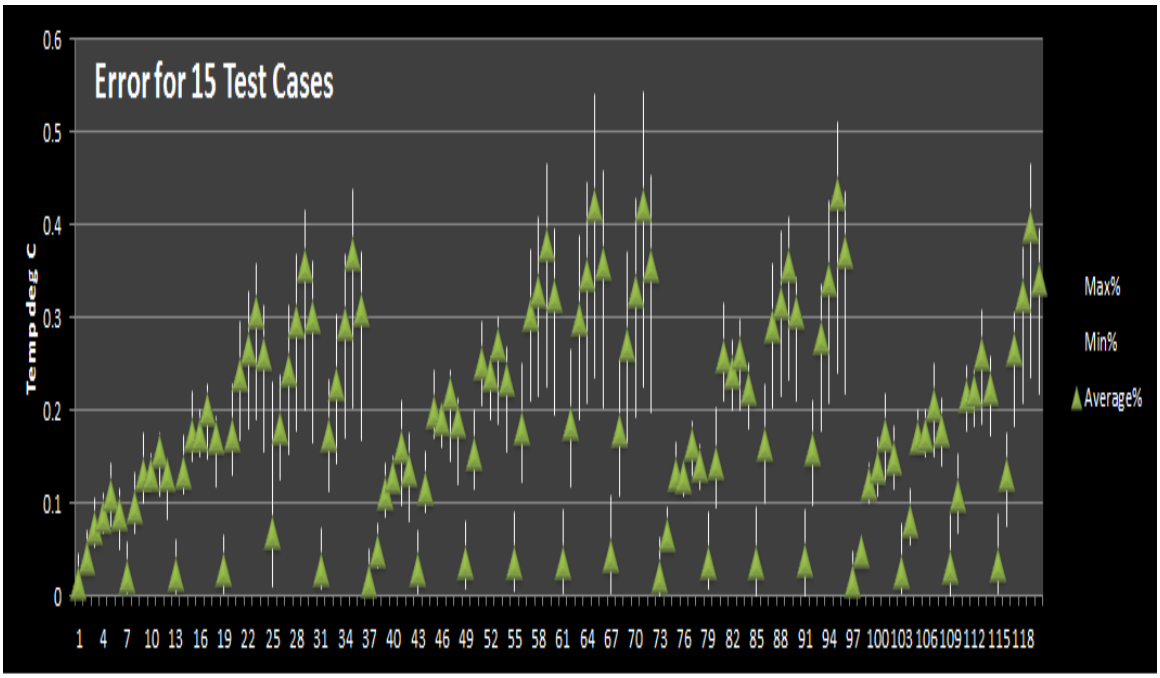

Figure 6.13. Error Percentage Chart.

6.2 Recurrent Network Architecture

The same test cases are chosen to test the performance of this architecture as were to test the performance of the feed forward architecture. The test cases were 5kW per rack for 60/

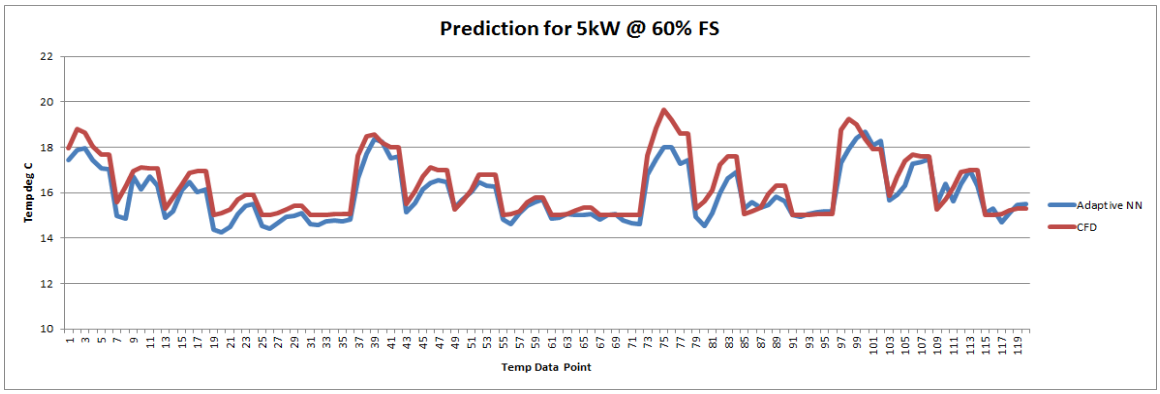

Figure 6.14. Prediction for 5kW at 60% CRAC fan speed.

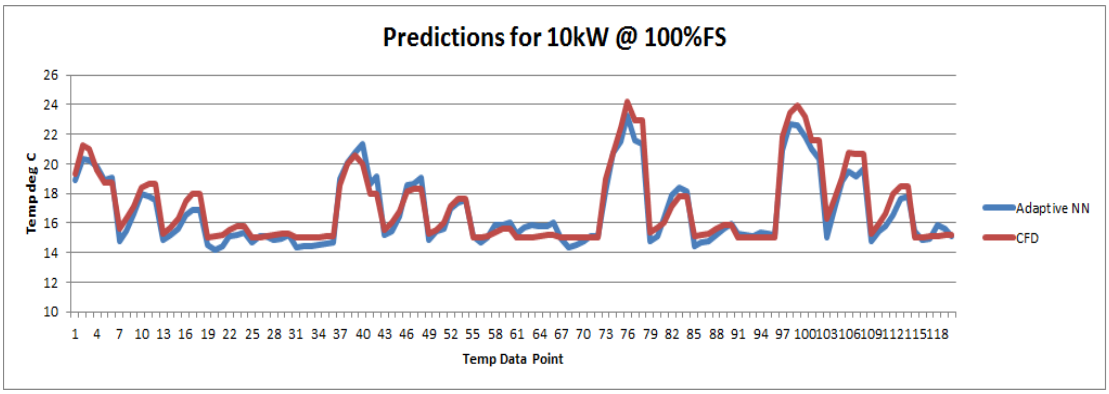

Figure 6.15. Prediction for 10kW at 100% CRAC fan speed.

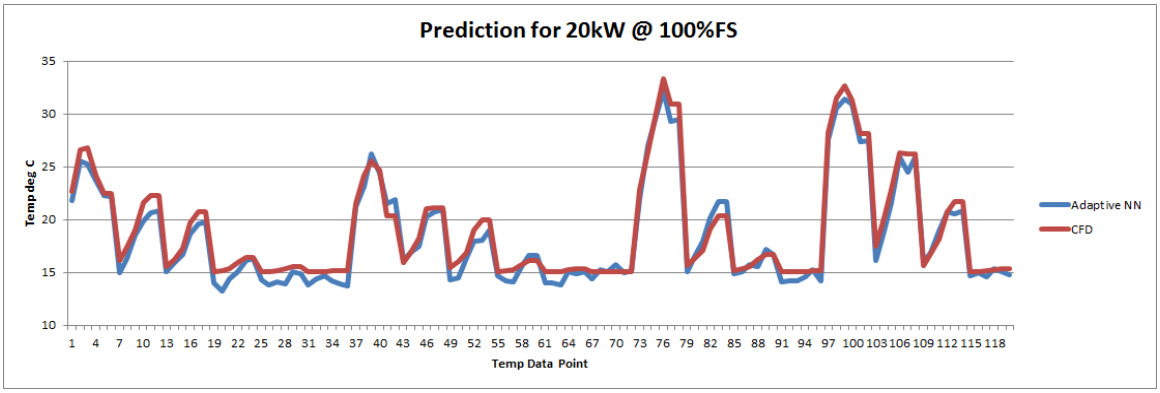

Figure 6.16. Prediction for 20kW at 100% CRAC fan speed.

Figure 6.17, 6.18 and 6.19 show the errors for each of the test cases. The average error percentage is 3.35% with a maximum error of 12.98% and a minimum of 0.009%. The average error, maximum error percentage and minimum error percentage is shown for 120 data points in figure 6.20

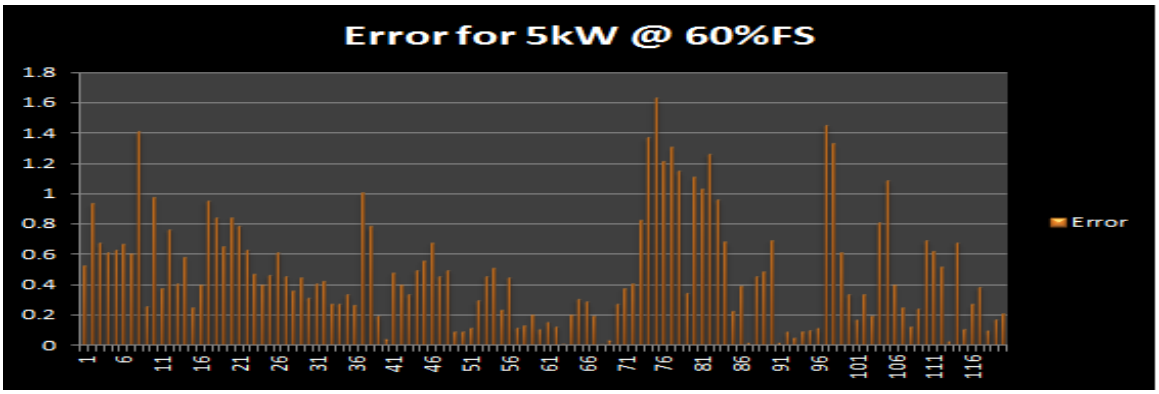

Figure 6.17. Errors for 5kW at 60% CRAC fan speed.

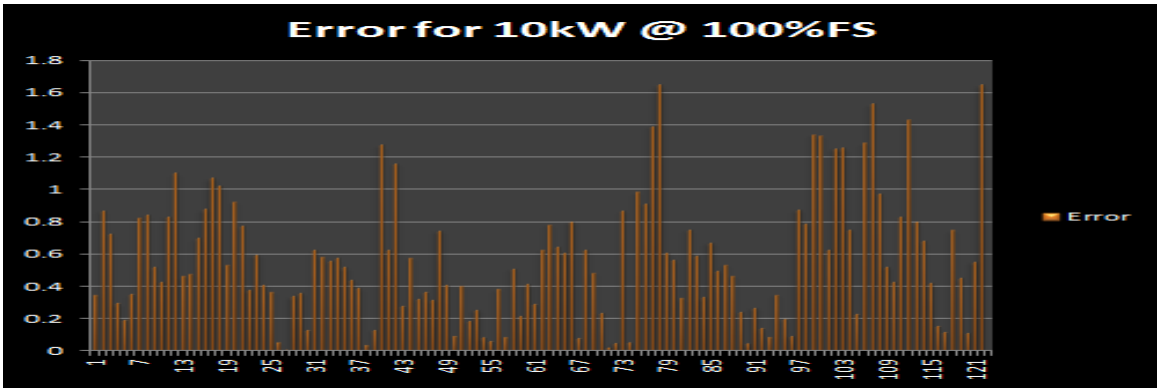

Figure 6.18. Errors for 10kW at 100% CRAC fan speed.

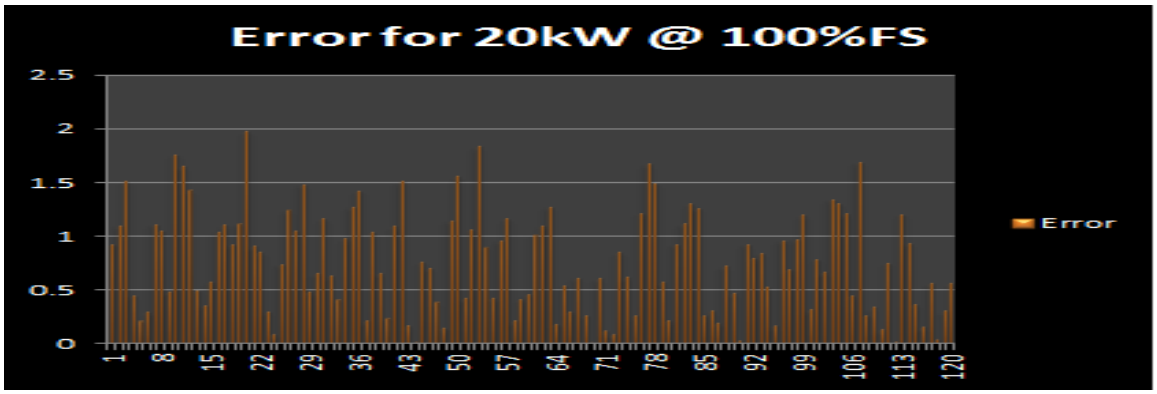

Figure 6.19. Errors for 20kW at 100% CRAC fan speed.

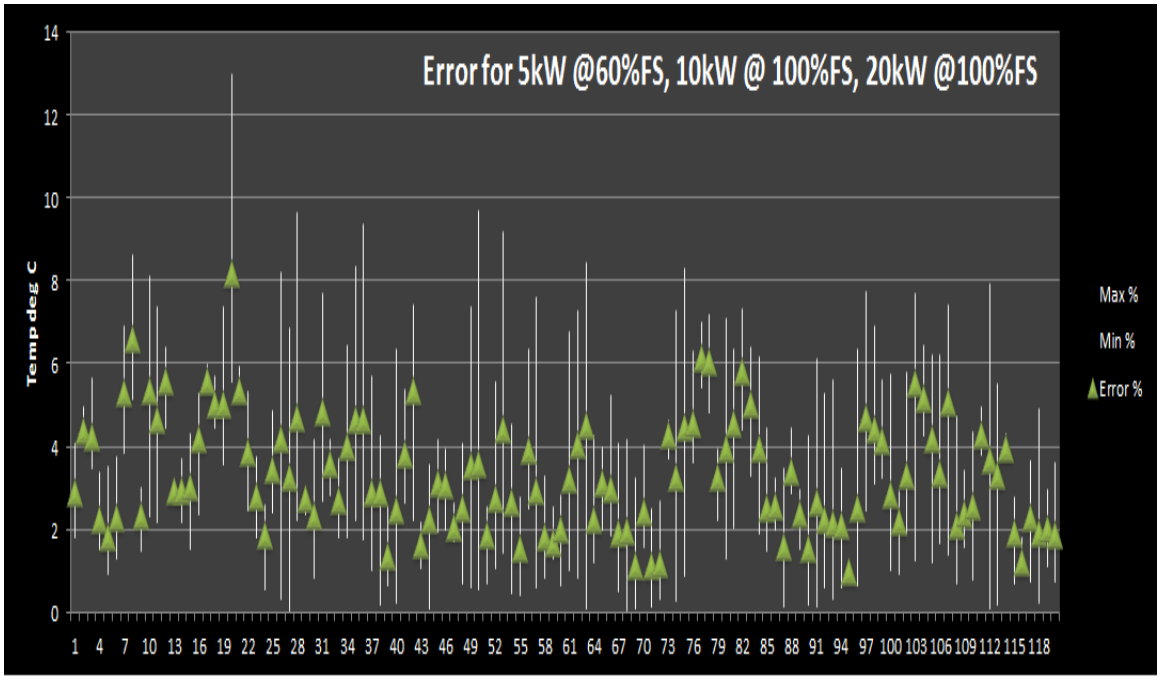

Figure 6.20. Error Percentage Chart.

CHAPTER 7

Conclusion and Future Work

7.1 Conclusion

An adaptive neural network has been developed that is shown to accurately predict the air temperatures in the data center room as a function of the server heat and CRAC fan speed and spatial position of the load banks. The network has been adaptively trained for a data set provided by CFD simulations. This adaptive training is advantageous as the network no longer needs to be trained on multitudes of data. With a minimal training set, the adaptive algorithm allows the network is incrementally update its weights with every pass.

Two different architectures have been presented with their merits. The feed-forward system showing a better accuracy of predictions and the recurrent system showing a slightly less accurate predictions but having the ability to include temporal data. The robustness and scalability of the feed forward system has been proved and can be used to learn a room with over 99% accuracy.

Further more, as this adaptive network has the flexibility to continually learn even after deployment. So it can receive real time data from a system and adapt itself further if need be. This feature is very useful when the network is used in a real time data center. Not only has it learned the room with minimal training, it can further adapt to new data fed to it.

The robustness of this system was verified when the network was tested for data that is had not been included in its data set. The accuracy shown in fig 16 proves the scalability of this network. The system can make predictions with an accuracy

of over 99%. By learning a new room via inputs from a plant model, the network's predictions proved that it is not limited to predictions with in its data set.

A strategy to save time and resources while maintaining a complex data center environment has been presented. The proposed system predicts temperatures for varying server configurations with a minimal error rate hence ensures the sustainability of this system. The various test cases and comparisons with the real time CFD data and black box model demonstrate the accuracy of system.

The neural network's speed and accuracy of predictions are two of its most promising claims to fame. The adaptive network takes this one step further, and shows that with minimal training the network can learn a new room and continually learn with out having to be retrained. The neural network can be a powerful tool in prediction of air inlet temperatures at key points in data center and this model can be extended to a real time cooling control model. Deployment of the proposed system is likely to save resources and the time invested in achieving optimal data center environment conditions.

7.2 Future Work

The proposed system is an initial attempt to use a new idea to predict optimal environments for servers and storage systems. However, this project has wider areas where which can be explored in the future. Some of them being :

1. Developing a deployment framework which will facilitate easy installation and operation of the proposed system in the targeted industries.
2. Deploying the neural network in a real time data center and monitoring the influence of the various environment variables on the results predicted.
3. Modify the current system to use different adaptive algorithms and further reduce the error in prediction rate.

4. Investigate any anomalies in prediction and investigate the cause and nature of the edge cases.
5. Incorporate the adaptive network into a dynamic controller[25] that can be deployed in a real time data center.

REFERENCES

[1] F. de Lorenzi and C. Vomel, "Neural Network-Based Prediction and Contol of Air Flow in a Data Center," *Journal of Thermal Science and Engineering Applications*, April 2012.

[2] D. Patnaik, M. Marwah, R. Sharma, and N. Ramakrishnan, "Sustainable Operation and Management of Data Center Chillers Using Temporal Data Mining," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 1305–1314. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557159

[3] "Rack and Power planning with HP Insight Control Power Management."

[4] Y. J. Emad Samadiani and F. Mistree, "The Thermal Design of a Next Generation Data Center: A Conceptual Exposition," *Journal of Electronic Packaging*, November 2008.

[5] ASHRAE, "Datacom Equipment Power Trends and Cooling Applications," ASHRAE, 2005.

[6] U. S. E. P. Agency, "Report to Congress on Server and Data Center Energy Efficiency."

[7] S. Alkharabsheh, B. Sammakia, S. Shrivastava, and R. Schmidt, "Utilizing practical fan curves in cfd modeling of a data center," in *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2013 29th Annual IEEE*, 2013, pp. 211–215.

[8] R. Steinbrecher and R. Schmidt, "Data Center Environments ASHRAE's Evolving Thermal Guidelines," *ASHRAE Journal*, vol. 53.

[9] M. J. E. Jr., "New ASHRAE Thermal Guidelines for Air and Liquid Cooling," *High Performance Computing, Networking Storage and Analysis, SC Companion:*, vol. 0, pp. 942–961, 2012.

[10] D. Livingstone, *Artificial Neural Networks: methods and applications*. Humana Press, 2008.

[11] G. Dreyfus, *Neural Networks: methodology and applications*. Springer, 2005.

[12] G. A. Works, "Neural Network Basics," in *SME Technological Papers*, 1989.

[13] P. Selvi, "Recognizing Handwritten Numerals Uing Multilayer Feed Forward Backpropagation Neural Networks," *International Journal of Computer Technology and Applications*.

[14] S. K. Chalup and A. D. Blair, "Incremental training of first order recurrent neural networks to predict a context-sensitive language," 2003.

[15] L. R. Medsker and L. C. Jain, *Recurrent neural networks: design and applications*. CRC Press, 2000.

[16] O. H and G. YP, "Recurrent neural network architecture with pre-synaptic inhibition for incremental learning." *Neural Networks : the Official Journal of the International Neural Network Society*, 2006.

[17] C. Ahn, "Robust stability of recurrent neural networks with ISS learning algorithm," *Nonlinear Dynamics*, vol. 65, no. 4, pp. 413–419, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11071-010-9901-5

[18] P. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[19] B. Perez-Sanchez, O. Fontenla-Romero, and B. Guijarro-Berdias, "An incremental learning method for neural networks in adaptive environments," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, 2010, pp. 1–8.

[20] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[21] L. Bruzzone and F. P. D, "An incremental-learning neural network for the classification of remote-sensing images," *Pattern Recognition Letter- Special Issue on Pattern Recognition in Practice.*

[22] T. L. Liang Ging, Chengliang Liu and Y. Fuqing, "Training feed-forward neural networks using the gradient descent method with the optimal stepsize," *Journal of Computational Information Systems*, vol. 8, no. 4, pp. 1359–1371, 2012X.

[23] C. J. B. Macnab, "A new robust weight update for multilayer-perceptron adaptive control," *Control Intell. Syst.*, vol. 35, no. 3, pp. 279–288, June 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1735647.1735659

[24] K. G. Sheela and S. N. Deepa, "Review on Methods to Fix Number of Hidden Neurons in Neural Networks," *Mathematical Problems in Engineering*, vol. 2013.

[25] Y. Kim and F. Lewis, "Neural Network Output Feedback Control of Robot Manipulators," *Robotics and Automation, IEEE Transactions on*, vol. 15, no. 2, pp. 301–309, 1999.

BIOGRAPHICAL STATEMENT

Vishok Amar Kumar was born in Bangalore, Karnataka, India in 1989. He received his Bachelors in Engineering in Mechanical Engineering from PES Institute of Technology Technology (Autonomous under Visvesvaraiah Technological University). He worked as a summer intern at Toyota, Bangalore, in 2009 and at Komatsu in 2011. He has been a part of the Electronics MEMS and Nanoelectronics Systems Packaging Center at the University of Texas at Arlington since 2011. His research interest includes neural networks and data center cooling applications.