Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Micro-Electronics Engineering

Investigation of Soft Neural Network Algorithm Implement to Analog Electronics Devices

By: Eyob Gedlie

Date Dec 31, 2018

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Investigation of Soft Neural Network Algorithm Implement to Analog Electronics Devices

By: Eyob Gedlie

A Thesis Submitted to Addis Ababa Institute of Technology, School of Electrical and Computer Engineering in Partials Fulfillment of the Requirements for the Degree of Masters of Science in Micro-Electronics Engineering.

Advisor: Dr. Eng' Getachew Alemu

Dec 31, 2018

Addis Ababa, Ethiopia

ADDISABABAUNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Investigation of Soft Neural Network Algorithm Implement to Analog Electronics Devices

By: Eyob Gedlie

Approved by the Examining Board:

_____	_____	_____
Chairman, Department's	Signature	Date
Graduate Committee		
Dr. Eng' Getachew Alemu	_____	_____
Advisor	Signature	Date
_____	_____	_____
External Examiner	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date

Abstract

The implementation of neural systems is presented in this paper. The thesis focuses on implementations where the algorithms and their physical support are tightly coupled. This thesis describes a neural network intelligent, application, soft-algorithm to implement to hardware electronics device. With the emerging of Integrated Circuit, any design with large number of electronic components can be squeezed into a tiny chip area with minimum power requirements, which leads to integration of innumerable applications so as to design any electronic consumer product initiated in the era of digital convergence. One has many choices for selecting either of these reconfigurable techniques based on Speed, Gate Density, Development, Prototyping, simulation time and cost. This thesis describes the implementation in hardware of an Artificial Neural Network with an Electronic circuit made up of Op-amps. The implementation of a Neural Network in hardware can be desired to benefit from its distributed processing capacity or to avoid using a personal computer attached to each implementation. The hardware implementation is based in a Feed Forward Neural Network, with a hyperbolic tangent as activation function, with floating point notation of single precision. The device used was an electronic circuit made with Op-amps The Proteus Software version 8.0 was used to validate the implementation results of the hardware circuit. The results show that the implementation does not introduce a noticeable loss of precision but is slower than the software implementation running in a PC.

Key Words: Algorithms, electronic, Neural Network, hardware and implementation.

Acknowledgement

First and for most I would like to thank the almighty God. I wish to express my deepest appreciation to my advisor, Dr. Eng' Getachew Alemu for his assistance, invaluable comments and excellent supervision throughout my research work. I would also like to thank classmates who were always willing to help and give valuable inputs. In addition, I would like to thank Mekdes Shiferaw for her help. I also extend my appreciation to Addis Ababa Institute of Technology, School of Electrical and Computer Engineering Laboratory staff members for their technical assistance.

Finally, I would like to express my sincere gratitude to my families especially to my father Mrs. Gedlie Lakew, my Mother Mamitu Bezabeh and my brother Biniyam Gedlie for their encouragement and support towards my thesis and academic career.

Table of Contents

List of Figures

Chapter One

Introduction

1.1. Background

The definition of a neural network, more properly referred to as an 'artificial' neural network (ANN), is provided by the inventor of one of the first neuron-computers, Dr. Robert Hecht-Nielsen. He defines a neural network as:"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs. "Or you can also think of Artificial Neural Network as computational model that is inspired by the way biological neural networks in the human brain process information. [1]

The basic computational unit of the brain is a neuron. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14}—10^{15} synapses. The diagram below shows a cartoon drawing of a biological neuron (left) and a common mathematical model (right).

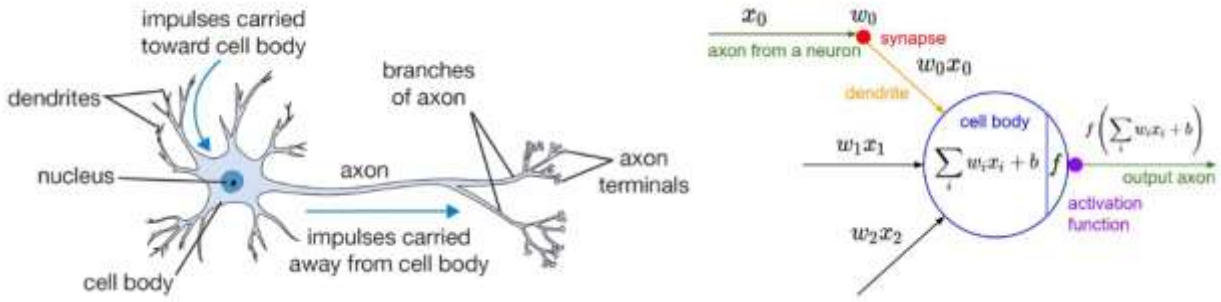


Figure 1 Biological neuron (left) and a common mathematical model (right)

The basic unit of computation in a neural network is the neuron, often called a node or unit. It receives input from some other nodes or from an external source and computes an output. Each input has an associated weight (w), which is assigned on the basis of its relative importance to other inputs. The node applies a function to the weighted sum of its inputs.

The idea is that the synaptic strengths (the weights w) are learnable and control the strength of influence and its direction: executor (positive weight) or inhibitory (negative weight) of one neuron

on another. In the basic model, the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon. In the computational model, we assume that the precise timings of the spikes do not matter, and that only the frequency of the firing communicates information. We model the firing rate of the neuron with an activation function (e.g. sigmoid function), which represents the frequency of the spikes along the axon. [2]

1.1.1. Neural Network Architecture

From the above explanation we can conclude that a neural network is made of neurons, biologically the neurons are connected through synapses where information flows (weights for out computational model), when we train a neural network we want the neurons to fire whenever they learn specific patterns from the data, and we model the fire rate using an activation function. [3]

I. Input Nodes (input layer): No computation is done here within this layer; they just pass the information to the next layer (hidden layer most of the time). A block of nodes is also called layer.

II. Hidden nodes (hidden layer): In Hidden layers is where intermediate processing or computation is done, they perform computations and then transfer the weights (signals or information) from the input layer to the following layer (another hidden layer or to the output layer).

III. Output Nodes (output layer): Here we finally use an activation function that maps to the desired output format (e.g. soft ax for classification).

IV. Connections and weights: The network consists of connections, each connection transferring the output of a neuron i to the input of a neuron j. In this sense i is the predecessor of j and j is the successor of i, each connection is assigned a weight W_{ij}.

V. Activation function: the activation function of a node defines the output of that node given an input or set of inputs. A standard computer chip circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input. This is similar to the behavior of the linear perceptron in neural networks. However, it is the nonlinear activation function that allows such networks to compute nontrivial problems using only a small number of nodes. In artificial neural networks this function is also called the transfer function.

VI. Learning rule: The learning rule is a rule or an algorithm which modifies the parameters of the neural network, in order for a given input to the network to produce a favored output. This learning process typically amounts to modifying the weights and thresholds.

1.1.2. Types of Neural Networks

There are many classes of neural networks and these classes also have sub-classes, here I will list the most used ones and make things simple to move on in this journey to learn neural networks.

A. Feed forward Neural Network

A feed forward neural network is an artificial neural network where connections between the units do not form a cycle. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

We can distinguish two types of feed forward neural networks:

Single-layer Perceptron

This is the simplest feed forward neural Network and does not contain any hidden layer, which means it only consists of a single layer of output nodes. This is said to be single because when we count the layers we do not include the input layer, the reason for that is because at the input layer no computations are done, the inputs are fed directly to the outputs via a series of weights. [2]

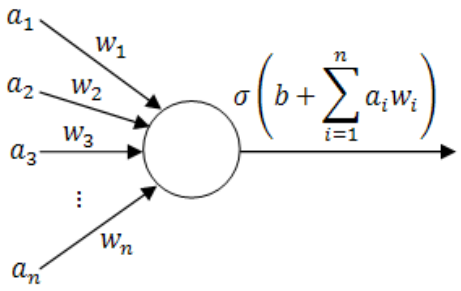


Figure 2 Simple perceptron

Multi-layer Perceptron (MLP)

This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the

subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function. [4]

Convolutional Neural Network (CNN)

Convolutional Neural Networks are very similar to ordinary Neural Networks, they are made up of neurons that have learnable weights and biases. In convolutional neural network (CNN, or ConvNet or shift invariant or space invariant) the unit connectivity pattern is inspired by the organization of the visual cortex, Units respond to stimuli in a restricted region of space known as the receptive field. Receptive fields partially overlap, over-covering the entire visual field. Unit response can be approximated mathematically by a convolution operation. They are variations of multilayer perceptron's that use minimal preprocessing. Their wide applications are in image and video recognition, recommender systems and natural language processing. CNNs requires large data to train on.

B. Recurrent neural networks

In recurrent neural network (RNN), connections between units form a directed cycle (they propagate data forward, but also backwards, from later processing stages to earlier stages). This allows it to exhibit dynamic temporal behavior. Unlike feed forward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegment, connected handwriting recognition, speech recognition and other general sequence processors.

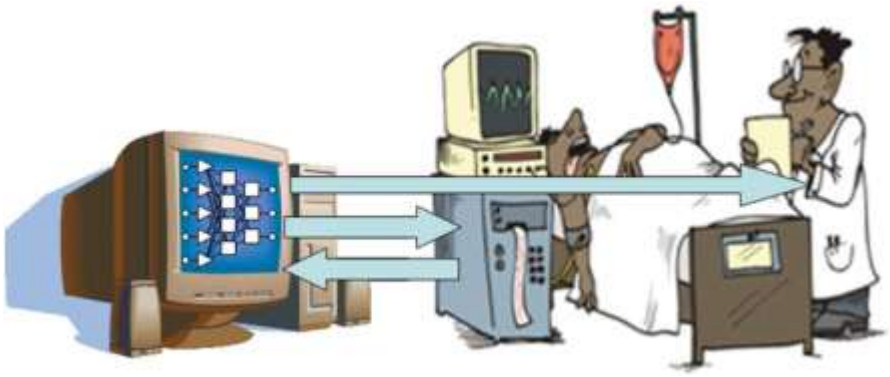
1.1.3. Neural Network Applications:

The ANN has seen an explosion of interest over the last few years and is being successfully applied across an extra ordinary range of problem domains in the area such as Medical, Aerospace, Image compression, stock Exchange Prediction etc.

Applications in Medical Field:

Medicine has always benefited from the forefront of technology. Technology advances like computers, lasers, ultrasonic imaging, etc. have boosted medicine to extraordinary levels of

achievement. Artificial Neural Networks (ANN) is currently the next promising area of interest. It is believed that neural networks will have extensive application to biomedical problems in the next few years. Already, it has been successfully applied to various areas of medicine, such as diagnostic systems, biochemical analysis, image analysis, and drug development. [4]



Diagnostic systems

ANNs are extensively used in diagnostic systems. They are normally used to detect cancer and heart problems. The benefit of using ANNs is that they are not affected by factors such as fatigue, working conditions and emotional state.

Biochemical Analysis

ANNs are used in a wide variety of analytical chemistry applications. In medicine, ANNs have been used to analyses blood and urine samples, track glucose levels in diabetics, determine ion levels in body fluids, and detect pathological conditions such as tuberculosis.

Image analysis

ANNs are used in the analysis of medical images from a variety of imaging modalities. Applications in this area include tumor detection in ultra-sonograms, classification of chest x-rays, tissue and vessel classification in magnetic resonance images (MRI), determination of skeletal age from x-ray images, and determination of brain maturation.

Drug development

ANNs are used as tools in the development of drugs for treating cancer and AIDS. ANNs are also used in the process of modeling biomolecules.

Modeling and Diagnosing the Cardiovascular System

Neural Networks are used experimentally to model the human cardiovascular system. Diagnosis can be achieved by building a model of the cardiovascular system of an individual and comparing it with the real time physiological measurements taken from the patient. If this routine is carried out regularly, potential harmful medical conditions can be detected at an early stage and thus make the process of combating the disease much easier. [4]

A model of an individual's cardiovascular system must mimic the relationship among physiological variables (i.e., heart rate, systolic and diastolic blood pressures, and breathing rate) at different physical activity levels. If a model is adapted to an individual, then it becomes a model of the physical condition of that individual. The simulator will have to be able to adapt to the features of any individual without the supervision of an expert. This calls for a neural network.

Another reason that justifies the use of ANN technology is the ability of ANNs to provide sensor fusion which is the combining of values from several different sensors. Sensor fusion enables the ANNs to learn complex relationships among the individual sensor values, which would otherwise be lost if the values were individually analyzed. In medical modeling and diagnosis, this implies that even though each sensor in a set may be sensitive only to a specific physiological variable, ANNs are capable of detecting complex medical conditions by fusing the data from the individual biomedical sensors. [4]

This model could be used to monitor employees in hazardous environments like fire-fighters. The system could be used to determine whether firemen have recovered sufficiently from the last inhalations of smoke to be allowed to enter smoke-filled environments again.

The advantages that such a system can offer are obvious. People can be checked for heart diseases quickly and painlessly and thus detecting any disease at an early stage. Of course, the system doesn't eliminate the need for doctors since a human expert is more reliable.

Pattern Recognition of Pathology Images

Pathology is an imaging technique in medicine which deals with the nature of disease (structural and functional changes in tissue). Its need of color and high resolution makes the use of digital image technology very difficult to implement.

Pattern recognition is an idea of classifying input data into identifiable classes by use of significant feature attributes of the data (sample), where the feature attributes are extracted from a background of irrelevant detail.

Neural Networks are used in pattern recognition because of their ability to learn and to store knowledge. Because of their 'parallel' nature can achieve, ANNs can achieve very high computation rates which is vital in application like telemedicine.

Instant physician

An application developed in the mid-1980s called the "instant physician" trained an auto associative memory neural network to store a large number of medical records, each of which includes information on symptoms, diagnosis, and treatment for a particular case. After training, the net can be presented with input consisting of a set of symptoms; it will then find the full stored pattern that represents the "best" diagnosis and treatment.

Applications in Stock Exchange Prediction:

The prediction accuracy of neural network has made them useful in making a stock market prediction. For large business companies, making predictions for stock exchange is common. This is by using parameters such as current trends, political situation, public view and economist’s advice. We can also use neural network in currency prediction, business failure prediction, debt risk assessment and credit approval.

Applications in Aerospace:

Technologies based on neural networks are currently being developed which may assist in addressing a wide range of complex problems in aeronautics. The review indicates that the proper

utilization of this technology offers a feasible approach to help meet current and future technological needs. This might include, but is not limited to, the following: the implementation of active control devices to harness or suppress unsteady aerodynamic effects such as dynamic stall on helicopter rotor blades; the parallel data processing of 10s to 1000s of sensors either for actuation and control or for system and component 'health' monitoring and fault detection; the development of simulators and control algorithms for severe, 'unsteady', six degree-of-freedom vehicle maneuvers where the linearized equations of motion do not adequately describe the vehicle dynamics; and the requirement to monitor these sensors, simulate the maneuvers and instigate control all on a time-scale which must be faster than the real-time phenomena. Clearly, neural networks alone will not solve these problems. However, neural networks provide a practical approach for determining solutions of complex nonlinear problems such as equations of motion, for the parallel processing of 1000s of sensors, and this can be achieved with the required computational speed. [4]

1.2. Statement Problem

Nowadays, computers are used to implement artificial Neural Networks which do the brain like functions. Using neural network which is an artificial representation of the human brain it is possible to simulate its learning process. An artificial neural network has an information processing system that could have certain performance features which is similar with biological neural networks. Artificial neural networks are developed using mathematical models of human cognition or neural biology, this are basically software tools representing the mathematical models in the form of artificial intelligent algorithms. Even though there are trials of developing the software neural network algorithms by hard ware, in this aspect there are many things to do further. In this thesis, the focus is on hardware representation of neural network algorithms. The hardware is not done yet properly and need further investigation. The aim of a trial contribution additional to the exiting, to propose a way mechanisms in which the already well-established software algorithm be represented by hardware system.

1.3. Objectives

1.3.1. General objective

- The general objective of this thesis is to design and implement the neural network with hardware electronic circuit, by using elementary circuit element such as resistor, Op-amp and diode

1.3.2. Specific objective

- The specific objectives of this thesis are
  - ❖ To thoroughly describe the theory behind neural networks
  - ❖ To determine the basic dynamics of neural network training from the mathematical model
  - ❖ To design the circuit model of the soft neural network
  - ❖ To compare the function of software neural network and the electronic hardware circuit.

1.4. Significance of the study

There are few researches which were done on artificial neural network in our country, Ethiopia. This thesis puts its own sign for the society. The fact is human being need assistance from robots. Robots are used in an industry to reduce human labor. In motor industries robots are used to assemble cars and another vehicle. Artificial neural network is useful to control robots. Robots work with artificial neural network. This thesis contributed how to substitute software based artificial network with hardware.

1.5. Scope and limitation

The scope of this study is to prepare artificial neural network and designing and simulate the hardware that can be used for artificial neural network and simulate the neural network. On top of that the designed hardware was analyzed. The circuit for the artificial neural network was constructed and checked using software. In addition, the function of the software neural network and hardware representation were compared.

Chapter Two

Literature Review

2.1. Literature survey of hardware implementations of neural networks

In this chapter, review of work carried out is presented in the form of literature survey. This paper presents a review on implementation issues of Artificial Neural Networks. Implementation models and various properties of the artificial neurons are discussed. We have surveyed various research contributions in the area of Neural Hardware. The review of implementation for neural hardware is divided into three broad categories – Analog, Digital and Hybrid architecture.

By going through the literature, the topic of the present study has been identified. Artificial neural networks and fuzzy systems have emerged as profitable tools for the control of complex industrial processes. The design of an ANN in hardware for real time applications involves the use of training algorithms and also some important factors like: the precision of inputs and weights, the computational function of a neuron and storage of weights.

Research work in the area of hardware implementation of Artificial Intelligence (AI) systems gained momentum in 80's. Research papers in FPGA implementation of AI systems started appearing in early 90's. The important research works available in the literature on the hardware implementation of neural network is outlined in the following sections. [5]

2.1.1. Hardware Implementation of Neural Networks

There are numerous chips available for building hardware circuits for the realization of neural networks and these have fixed functionality. Due to the wide range of sizes of field programmable devices, they can be used to realize much larger digital circuits than what standard chip can realize, to meet the requirements of a specific application. These devices have the ability of reprogramming to make any corrections during testing. The simplification of hardware saves cost and improves system reliability. The development of sophisticated field-programmable devices has led to the realization of different digital hardware over the past few years and provided a powerful means to implement artificial neural network with the high degree of parallelism. [6]

Various approaches for the hardware implementation of neural networks have been described by Collins *et al.,* 1991, Van Daalen *et al.*, 1991 and Ferrucci, 1994. Reay *et al*., 1994 have implemented a new architecture called Cerebellar Model Articulation Controller (CMAC) for neural networks and have concluded that the hardware architecture of CMAC implemented on FPGA is suited for only a specific application. Paul Morgan *et al.,* 1994 have proposed the hardware implementation of a probabilistic RAM based neural network architecture, named HyperNet, in terms of system training speed, size, and component cost. Himanshu S. Mazumdar, 1995 has proposed a new hardware for the implementation of multilayer feed-forward neural network with neurons working in parallel and programmable, and the layers working in sequential mode. The hardware and software implementations of neural network architectures have been analyzed by Blake *et al*., 1997. Dan Hammerstrom, 1998 has focused on the differences between digital and analog design techniques of neural networks with cost performance trade-offs. A survey of implementations of artificial neural networks on several massively parallel hardware has been made by Udo Seiffert, 2002. The use of reconfigurable feature of field-programmable devices in neural networks has been reported by Erdogan *et al*., 1993, Lysaght *et al*., 1994, Yamina Tarıght and Michel Hubin 1998 and Seok Bae Yun *et al.*, 2002. [7]

Schoenauer *et al*., 1998 have analyzed the importance of digital neuro hardware in terms of system architecture, degree of parallelism, typical neural network partition per processor, inter-processor communication network and numerical representation. The hardware realization of neuron function and the method of parallelism have been attempted by several researchers (Cox and Blanz, 1992, Speckmann *et al.,* 1993, Hon Keung Kwan and Chuan Zhang Tang, 1993, Ayala *et al*., 2002, Acosta Nelson and Tosini Marcelo, 2002 and Wang Qinruo, 2003). Gilberto Contreras *et al*., 2003 have concluded that the hardware implementations of ANN offer faster execution than general purpose microprocessors by taking advantage of reusable modules, parallel processing and specialized computational components. The highest degree of parallelism is required in the connections of a neuron from all the neurons of the previous layer wherein several calculations are to be made at the same time. The neuron needs as many multipliers as the number of connections from the previous layer. This method is fast but inflexible, because the architecture of a neuron has to be changed for a different sizes of network.

The time multiplexing of the interconnections between the neurons has been proposed by Ossoinig, 1995 and Acosta Nelson and Tosini Marcelo, 2002. Ossoinig, 1995 has concluded that the node parallelism gives better performance than the link parallelism. The neuro processor developed by Cheang, 2003 executes the Kohonen network algorithm using the synapse-parallel approach. In the link parallelism discussed by Ma Xiaobin *et al*., 2003, the data between two layers is transferred serially. A possible solution to the weight storage problem has been highlighted by Moritoshi Yasunaga *et al*., 1990 and it is based on a technique called as 'most significant weights' philosophy. The various aspects of hardware requirements must be analyzed in digital implementation of neural networks in order to achieve high performance and density (Cesare Alippi and Meyer E. Nigri, 1991). These hardware requirements refer to the data representation (fixed or floating point) and their accuracy, the way arithmetic computation is carried out and activation function realization.

The common ways of implementing activation function are look-up table method and approximation method. The look-up table method attempted by Coric *et al*., 2000, Acosta and Toshini, 2002 needs a lot of memory space for look-up table entries, but the approximation method suggested by Tommiska, 2003 requires less amount of hardware components leading to good performance and less gate requirement. Based on the survey of FPGA implementation of neural networks made by Zhu and Sutton, 2003 and Liu and Liang, 2005, the logical XOR networks occupy almost 50% of Configurable Logic Block (CLB) slices of the Virtex device with a maximum clock rate of 10 MHz for 16-bit fixed point arithmetic operations. The hardware implementation of neural networks on FPGA with fixed point arithmetic has been proposed and carried out by Eickhoff *et al*., 2006. But the fixed point arithmetic has limiting factor for most applications due to two major problems i.e. the dynamic range of computation and the inflexibility to customize the hardware circuit. These limitations can be overcome by using floating point arithmetic operations as suggested by Hok, 2003 and Xiaoguang *et al*., 2006. [8]

On-chip learning is necessary for most of the real time applications of neural networks. The on-chip learning of feedforward neural network is generally performed by backpropagation algorithm. Several authors, Holt and Baker, 1991, Hikawa, 1995, Girau and Tisserand, 1996, Hikawa, 2000, Nichols *et al*., 2002 and Gadea *et al*., 2005 have discussed the importance of on-chip learning of feedforward neural networks. Hikawa, 1995 and 2000 has suggested a pulse mode arithmetic operation and Perez-Uribe and Sanchez, 1997 have presented a FPGA design of an unsupervised

neural network with adaptable structure which uses an adaptive heuristic critic learning. Gadea *et al*., 2000 and 2005 have proposed new pipelined design based on systolic array implementation to perform online backpropagation algorithm. Czauderna and Seiffert, 2004 have proposed a Message Passing Interface (MPI) based parallel variant of backpropagation algorithm which was run on a number of parallel computer architectures. [9]

VLSI implementations of RBF neural networks have been attempted by Seth Wolpert *et al*., 1992, Choi *et al*., 1994, Aberbour and Mehrez, 1998 and Brassi and Bako, 2007. The hardware implementation of RBF network suggested by Aberbour and Mehrez, 1998 is based on DDA (Dynamic Decay Adjustment) model complying fully with the mathematical formulation of the algorithm. The training of an RBF neural network has been performed by selecting the centers of hidden layer neurons and then by estimating the output layer weights (Haykin 1994). The centers of the hidden layer neurons of an RBF neural network are selected by using some clustering algorithms like *k*-means (Moody and Darken, 1989 and Sing *et al*., 2003) and fuzzy c-means (Zhu and He, 2006). Pereira *et al*., 2000 have suggested the use of recursive *k*-means clustering algorithm for updating the hidden layer centers and recursive least square algorithm for updating the weights of the output layer neurons in RBF network. Fernandez-Redondo *et al*., 2006 have also suggested the use of gradient descent method for updating the output layer weights. The hardware implemented for centroid updation by Liu *et al*., 2005 has been executed on an Altera's Stratix device with a NiosII 50MHz CPU and 16MB of SDRAM and takes approximately 2 million clock cycles for calculating all centroids in parallel. [10]

2.1.2. Implementation for Neural Hardware

Implementation for neural hardware is divided into three broad categories –

a. Analogue Hardware implementations
b. Digital Hardware implementations
c. Hybrid Hardware implementation

In recent years, many approaches have been proposed for implementing the Artificial Neural Networks. An investigation on neural network hardware devices proposed in the literature reveals that the following block level architectural representation is suitable for describing almost all neuron-chips and neuron-computer processing elements. [11]

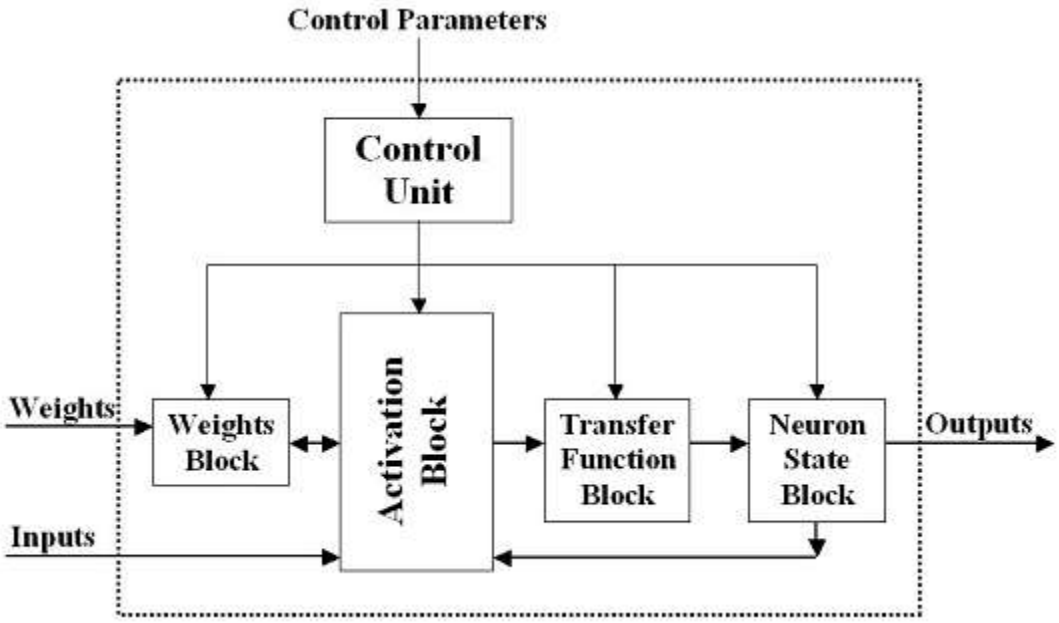


Figure 3: Block level architectural representation of a neuron-chip or a neuron-computer-processing element

The activation block, which evaluates the weighted sum of the inputs, is always on the neuron-chip. Other blocks, i.e. the Neuron State Block, Weights Block, and the Transfer Function Block may be on the chip, or off the chip. A host computer may perform some of these functions and computations. The Control Unit that is always on-chip controls the data flow between these blocks. The control parameters are used for controlling the chip by a host. The data flow is such that the weights from weights block and the inputs from outside or from the outputs are multiplied and the products are summed in the activation block, then the outputs are obtained in the neuron state block from the transferred sum of products.

For multiplayer perceptron and Hopfield network the transfer function may be a threshold, linear, ramp or sigmoid function, for Boltzamann machine it is a threshold function having some noise included. Neuron states and weights can be stored in digital form, or in analog form. Weights can be loaded statically, only once, before the activation computation, or they can be updated dynamically by the host or by the activation block in a learning phase, while activation steps are being performed. [12, 13]

2.1.3. Neural Network Hardware Classification

Following attributes can be used to classify and compare the characteristics of neural network hardware.

Type of device

The device of interest may be:

I. a neuron-chip or a neuron-computer

II. a general purpose device or a special purpose device

The first attribute is used to indicate if the device is a neuron-chip or a neuron-computer. While a neuron-chip is a single chip, a neuron-computer is an architectural system built using basic blocks generally referred to as "processing elements", interconnected in a special manner. The second attribute indicates whether the device is a general purpose or a special purpose one. A general-purpose device can be used for implementing more than one algorithm such as Back propagation, Hopfield, Kohonen, etc., whereas a special purpose device can be used for implementing only one type of algorithm.

Neuron Properties

Sub-attributes related with the properties of neurons in the device are:

i. The number of neurons

ii. Storage of neuron-state: on-chip/off-chip

iii. Neuron State: digital/analog

iv. Precision (in number of bits)

The number of neurons is directly stated. The storage of neuron state can be on-chip, or off-chip. If the neuron state is stored on the chip, then it can be stored in analog or digital form, with the analog form having the alternatives of keeping states in terms of voltage, frequency, etc. Precision is the number of bits used to represent neuron output value. If outputs are digital, the neuron state is considered to be kept in digital form. If outputs are transmitted to the host as analog voltage, the neuron state is said to be kept in analog form.

Weights

Sub attributes related with weights are:

i. Storage of weights: on-chip/off-chip
ii. Number of synapses
iii. Weights: analog/digital

Storage of weights can be on the chip, or off the chip. If the weights are stored on the chip, storage type can be digital or analog. If the storage type is analog, weights can be kept in terms of voltage, electrical charge or resistance values. If the storage type is digital, weights are either stored before the activation, and they are not modified later (static) or they are allowed to be updated. [14]

Activation characteristics

Sub attributes determining the activation characteristics, which require the sum of products computation are:

i. Computation: digital/analog
ii. Activation block output: probabilistic/deterministic

The computation of activation is always performed on the chip in digital or in analog form. If the sum of products computation results is directly applied to the transfer function, it is deterministic. Sometimes, a noise factor can be introduced to the output of the activation block or to the inputs of the chip. In this case, looking at the activation value, the activation block output may be affected by the noise value, which means the result becomes probabilistic.

Transfer function characteristics

Sub attributes that determine the transfer function characteristics are:

i. Transfer function: on-chip/off-chip
ii. Transfer function: analog/digital
iii. Threshold/look-up table/computation

The transfer function can be performed on the chip or off chip, and it can be analog or digital. It is digital then a look up table can be used, a threshold comparison can be done, or computation can be directly performed. If it is analog, an electronic circuitry, such as op-amp, can be used for this purpose. [15]

Back-Propagation Algorithm: - Multi-layered perceptron's can be trained using the back-propagation algorithm described next. Goal: To learn the weights for all links in an interconnected multilayer network. We begin by defining our measure of error: [16]

$$E(W) = \frac{1}{2}\sum d \sum K\,(t_{Kd} - O_{Kd})^2$$

k varies along the output nodes and d over the training examples. The idea is to use again a gradient descent over the space of weights to find a global minimum (no guarantee).

Algorithm:

1. Create a network with n in input nodes, n hidden internal nodes, and n out output nodes.

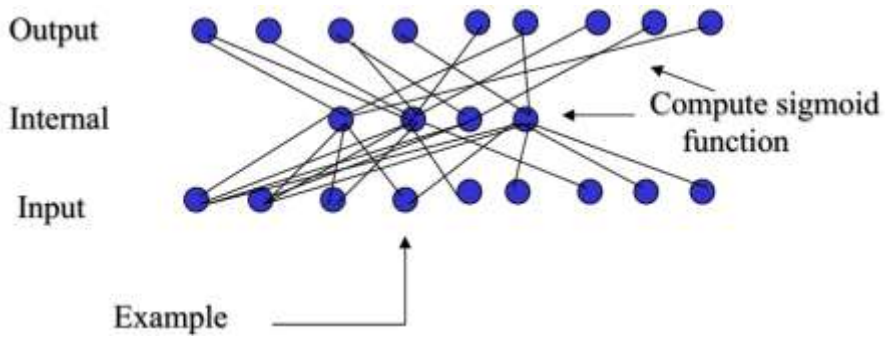
2. Initialize all weights to small random numbers.

3. Until error is small do:

    For each example X do

    • Propagate example X forward through the network
    • Propagate errors backward through the network

Forward Propagation Given Example X, compute the output of every node until we reach the output nodes:



Backward Propagation

A. For each output node k compute the error:

    $\delta_k = O_k\,(1\text{-}O_k)\,(t_k - O_k)$

B. For each hidden unit h, calculate the error:

    $\delta_h = O_h\,(1\text{-}O_h)\,\Sigma k\; W_{kh}\,\delta_k$

C. Update each network weight:

$W_{ji} = W_{ji} + \Delta W_{ji}$

Where $\Delta W_{ji} = \eta\ \delta_j\ X_{ji}$ (W_{ji} and X_{ji} are the input and weight of node i to node j)

Information flow

The way that weights are inputted to the activation block, and the way that the computation results are outputted into the transfer function block determine the general information flow, which is a property, used for comparison of the devices. If neuron states and weights are kept off the chip, the information flow type becomes more important. Among the block, data can be flowing using systolic array, direct connection, broadcast broad-cast or pipelining techniques. [17]

Perceptron Learning

Learning a perceptron means finding the right values for W. The hypothesis space of a perceptron is the space of all weight vectors. The perceptron learning algorithm can be stated as below.

1. Assign random values to the weight vector

2. Apply the weight update rule to every training example

3. Are all training examples correctly classified?

    A. Yes. Quit

    B. No. Go back to Step 2.

There are two popular weight update rules.

i) The perceptron rule, and

ii) Delta rule

The Perceptron Rule

For a new training example $X = (X_1, X_2 \ldots X_n)$, update each weight according to this rule:

$W_i = W_i + \Delta W_i$

Where $\Delta W_i = \eta(t - O)x_i$

t: target output

o: output generated by the perceptron

η: constant called the learning rate (e.g., 0.6)

Comments about the perceptron training rule:

•If the example is correctly classified the term (t-o) equals zero, and no update on the weight is necessary.

• If the perceptron outputs –1 and the real answer is 1, the weight is increased.

• If the perceptron outputs a 1 and the real answer is -1, the weight is decreased.

• Provided the examples are linearly separable and a small value for η is used, the rule is proved to classify all training examples correctly (i.e., is consistent with the training data).

Deep learning is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design, where they have produced results comparable to and in some cases superior to human experts.

Deep learning models are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains, which make them incompatible with neuroscience evidences.

Most modern deep learning models are based on an artificial neural network, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in Deep Belief Networks and Deep Boltzmann Machines.

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own. (Of course, this does not completely obviate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction.) [18]

The "deep" in "deep learning" refers to the number of layers through which the data is transformed. More precisely, deep learning systems have a substantial credit assignment path (CAP) depth. The CAP is the chain of transformations from input to output. CAPs describe potentially causal connections between input and output. For a feed forward neural network, the depth of the CAPs is that of the network and is the number of hidden layers plus one (as the output layer is also parameterized). For recurrent neural networks, in which a signal may propagate through a layer more than once, the CAP depth is potentially unlimited. No universally agreed upon threshold of depth divides shallow learning from deep learning, but most researchers agree that deep learning involves CAP depth > 2. CAP of depth 2 has been shown to be a universal approximate in the sense that it can emulate any function. Beyond that more layers do not add to the function approximate ability of the network. The extra layers help in learning features.

Deep learning architectures are often constructed with a greedy layer-by-layer method. Deep learning helps to disentangle these abstractions and pick out which features improve performance.

For supervised learning tasks, deep learning methods obviate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures that remove redundancy in representation.

Deep learning algorithms can be applied to unsupervised learning tasks. This is an important benefit because unlabeled data are more abundant than labeled data. Examples of deep structures that can be trained in an unsupervised manner are neural history compressors and deep belief networks. [18]

Types of Implementation:

Classify the implementation types of the hardware solutions is a controversial task, as can be seen from the examples present in the literature. For example, in (Aybay et al., 1996) a classification is proposed which divides the hardware solutions in Neuron-chips and Neuron-computers having both subcategories of special and general purpose. Another classification, although more general since it is meant for ANN hardware components, is given in (Caviglia), where the main types are: neuron-computers, personal computer accelerators, chips, cell libraries and embedded microcomputers. [19]

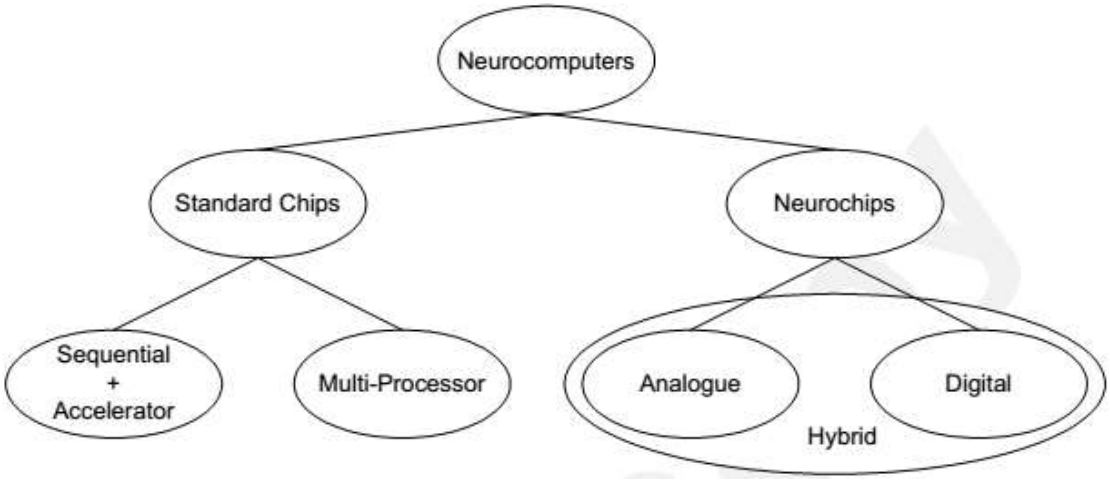


Figure 4 Neural network Hardware categories

Chapter Three

Hardware Implementation of ANN and SNN Model

Solutions with ANNs (Artificial Neural Networks) reach better results in the implementation phase with specific hardware than the most common implementation using a personal computer or workstation. The existence of these solutions in hardware is of extraordinary importance for areas as Applied Sciences and Health Sciences. The ANNs are parallel distributed systems, since they have the capacity to receive several inputs at the same time and to distribute these inputs in an organized manner. With this architecture, the information stored and shared in all the processing units of the ANNs improve the performance and the reliability in the systems implemented.

The implementation of ANN will have to supply a signal to the output of the system that will activate the coupled components, resembling the one that happens in the Biological Neural Network. [20] 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.

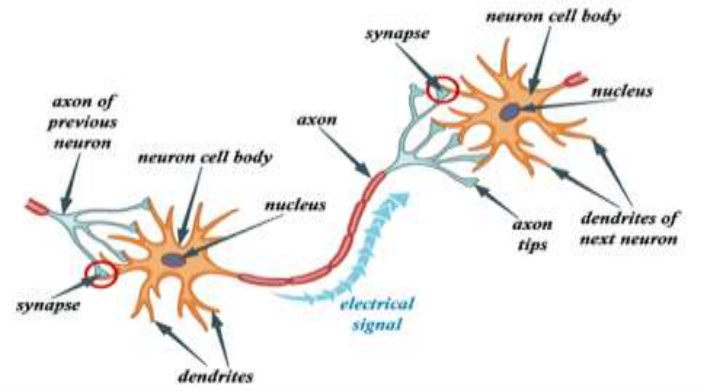


Figure 5: Biological neuron anatomy

source: *http://www.myreaders.info/html/soft_computing.html*

Artificial neural networks are among the most powerful learning models. They have the versatility to approximate a wide range of complex functions representing multidimensional input-output maps. Neural networks also have inherent adaptability, and can perform robustly even in noisy environments. [21] An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected simple processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. ANNs can process information at a great speed owing to their highly massive parallelism.

Artificial Neural Network A system which performs information processing. An ANN resembles or it can be considered as a generalization of mathematical model of human brain assuming that

- Information processing occurs at many simple elements called neurons.
- Signals are passed between neurons over connection links.
- Each connection link has an associated weight, which in a typical neural net multiplies the signal transmitted.

ANN is built with basic units called neurons which greatly resemble the neurons of human brain. A neural net consists of a large number of simple processing elements called neurons. Each neuron applies an activation function to its net input to determine its output signal. Every neuron is connected to other neurons by means of directed communication links, each with an associated weight.

Each neuron has an internal state called its activation level, which is a function of the inputs it has received. This can be compared with a bottle with a liquid. If we have a bottle and if we fill in the bottle with a liquid, and if we have an alarm to caution us when the level of the liquid is up to the neck of the bottle, then activation level also does the same thing as that of the alarming signal we receive.

As and when the neuron receives the signal, it gets added up and when the cumulative signal reaches the activation level the neuron sends an output. Till then it keeps receiving the input. So activation level can be considered as a threshold value for us to understand.

An Artificial Neuron is the basic unit of a neural network. A schematic diagram of a neuron is given below.

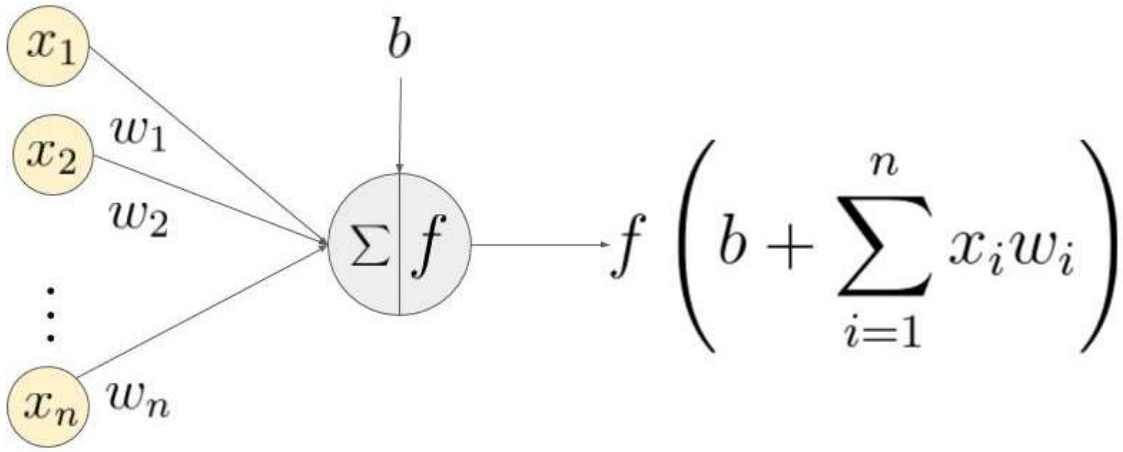


Figure 6: an example of a neuron showing the input (x_1-x_n), their corresponding weights(w_1-w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

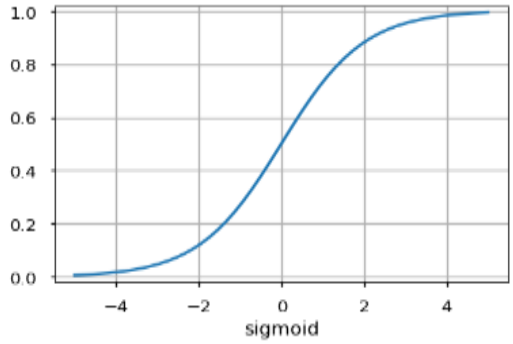
As seen above, it works in two steps – It calculates the weighted sum of its inputs and then applies an activation function to normalize the sum. The activation functions can be linear or nonlinear. Also, there are weights associated with each input of a neuron. [6]

Activation Functions

The activation function is used as a decision making body at the output of a neuron. The neuron learns Linear or Non-linear decision boundaries based on the activation function. It also has a normalizing effect on the neuron output which prevents the output of neurons after several layers to become very large, due to the cascading effect. There are three most widely used activation functions. [15]

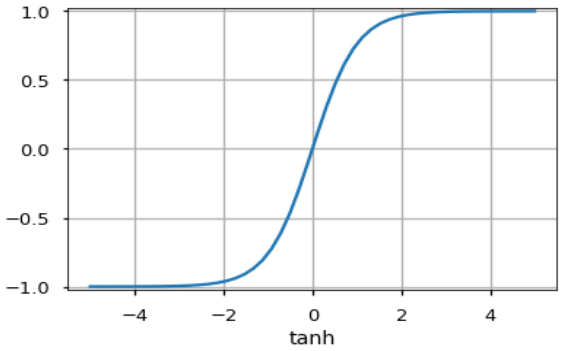
Sigmoid

It maps the input (X axis) to values between 0 and 1.



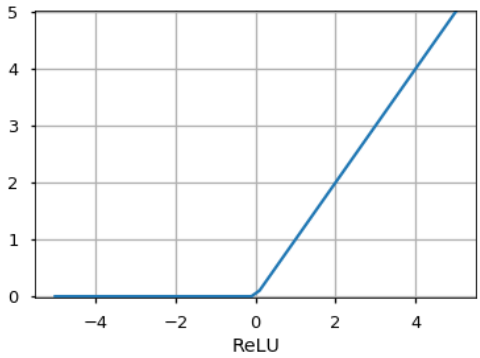
Tanh

It is similar to the sigmoid function but maps the input to values between -1 and 1.



Rectified Linear unit RlEU

It allows only positive values to pass through it. The negative values are mapped to zero.



There are other functions like the Unit Step function**,** leaky ReLU**,** Noisy ReLU**,** Exponential LU etc. which have their own merits and demerits.

In general, a neural network is characterized by

- Pattern of connections between the neurons called its architecture
- Method of determining the weights on the connections called its training or learning algorithm
- Its internal state called its Activation function

The arrangement of neurons into layers and the connection patterns within and between layers is called the net architecture. A neural net in which the signals flow from the input units to the output units in a forward direction is called feed forward nets. The fully interconnected competitive net in which there are closed loop signal paths from a unit back to it is called a recurrent network.

Neural nets can also be classified based on the above stated properties. A neural net with only input layer and output layer is called single layer neural network. A neural network with input layer, one or more hidden layers and an output layer is called a multilayer neural network. A single layer network has limited capabilities when compared to the multilayer neural networks.

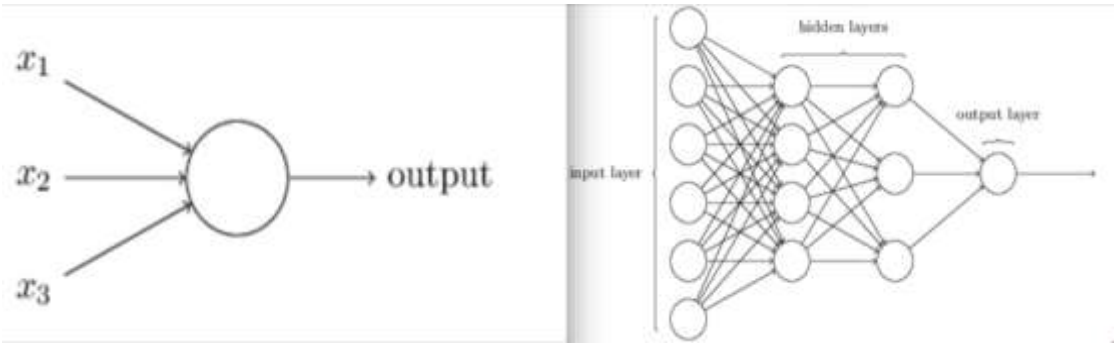


Figure 7 the left image is single layer neural network and right layer is the image of Multilayer neural network

There are a wide variety of networks depending on the nature of information processing carried out at individual nodes, the topology of the links, and the algorithm for adaptation of link weights. Some of the popular among them include: [4]

Perceptron: This consists of a single neuron with multiple inputs and a single output. It has restricted information processing capability. The information processing is done through a transfer function which is either linear or non-linear.

Multi-layered Perceptron (MLP): It has a layered architecture consisting of input, hidden and output layers. Each layer consists of a number of perceptron's. The output of each layer is transmitted to the input of nodes in other layers through weighted links. Usually, this transmission is done only to nodes of the next layer, leading to what are known as feed forward networks. MLPs were proposed to extend the limited information processing capabilities of simple perceptron's, and are highly versatile in terms of their approximation ability. Training or weight adaptation is done in MLPs using supervised back propagation learning.

Recurrent Neural Networks (RNN): topology involves backward links from output to the input and hidden layers. The notion of time is encoded in the RNN information processing scheme. They are thus used in applications like speech processing where inputs are time sequences data.

In addition to architecture, the method of setting the values of the weights called training is an important characteristic of neural nets. Based on the training methodology used neural nets can be distinguished into supervised or unsupervised neural nets.

For a neural net with supervised training, the training is accomplished by presenting a sequence of training vectors or patterns each with an associated target output vector. The weights are then adjusted according to a learning algorithm. For neural nets with unsupervised training, a sequence of input vectors is provided, but no target vectors are specified. The net modifies the weights so that the most similar input vectors are assigned to the same output unit. The neural net will produce a representative vector for each cluster formed. Unsupervised learning is also used for other tasks, in addition to clustering. [5]

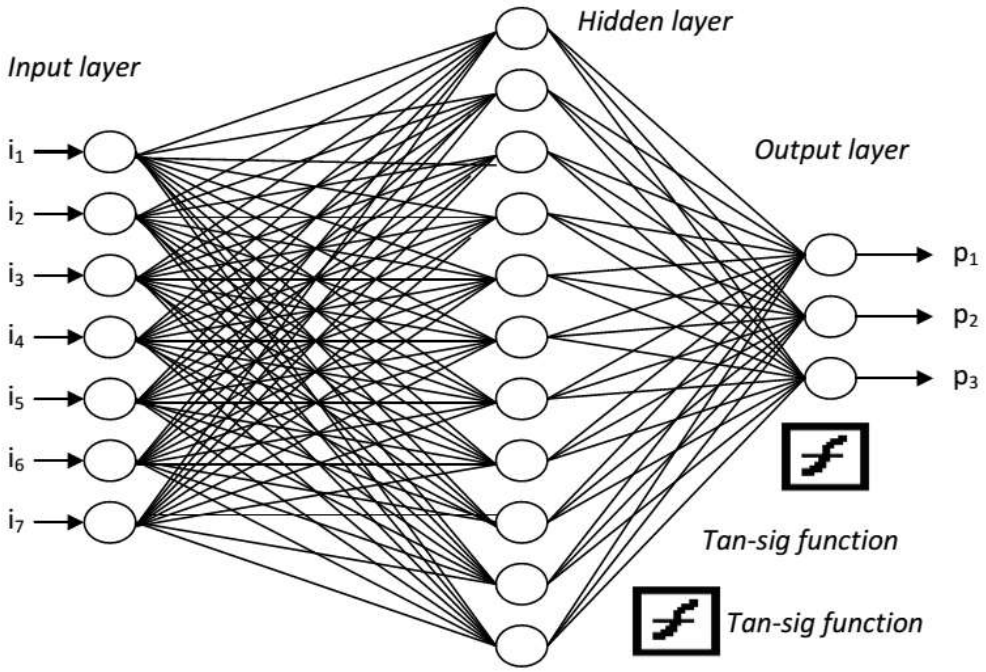


Figure 8 Architecture of a neural network

3.1. Implementation

A neural network may have different layers of neurons like input layer, hidden layer, and output layer. The input layer receives input data from the user and propagates a signal to the next layer called the hidden layer. While doing so it multiplies the weight along with the input signal. The hidden layer is a middle layer which lies between the input and the output layers. The hidden layer with nonlinear activation function increases the ability of the neural network to solve many problems than the case without the hidden layer. The output layer sends its calculated output to the user from which decision can be made.

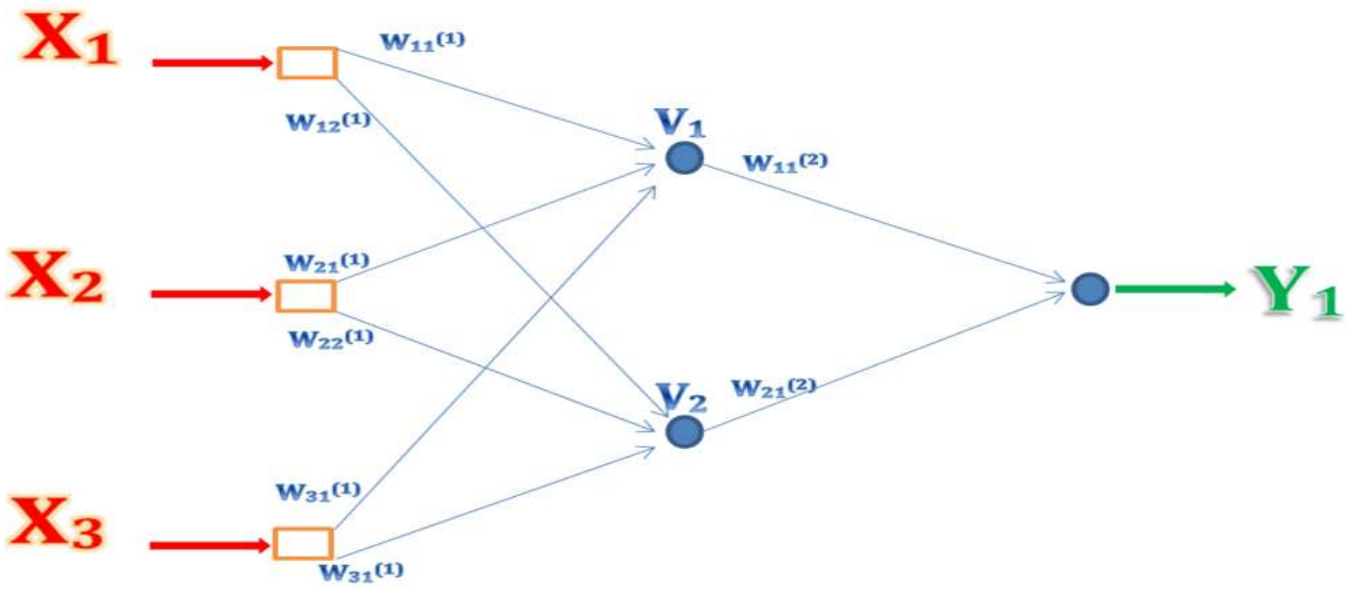
Neural network consists of an input layer, an output layer and a hidden layer. While a neural network is constructed, the number of neurons in each layer has to be fixed. The input layer will have neurons whose number will be equal to the number of features extracted. The number of neurons in the output layer will be equal to the number of pattern classes. The number of neurons in the hidden layer is decided by trial and error basis. With a minimum number of neurons in the hidden layer, the neural

network will be constructed and the convergence will be checked for. Then the error will be noted. The number of neurons for which the error is minimum, can be taken and will be checked for reduced error criterion.

The type of Neural Network used in this project was Feed Forward Neural Network, i.e. a network that allows only connections in the output direction. This option result of its bigger spreading at the level of the developed tools that allow the fast implementation of models. The network would have to receive a variable number of inputs and neurons. The number of neurons of the hidden layer allows us to adjust the size of the network to the complexity of the system being used.
During the project, several architectures were tested. The notation used defines the architecture in a simple form: they quantify the number of inputs, neurons in the hidden layer and neurons in the output layer, i.e., for the network there are three inputs, five neurons in the hidden layer and one neuron in the output layer.

3.2. Mathematical model of artificial neural network

To design the hardware, the first step is modeling the artificial neural network with mathematics. Artificial neural network is an interconnected group of artificial neurons that uses a mathematical model or computational model for information processing based on a confectioning approach to computation. The artificial neural network model is calculated as described below.



Where X is input layer

W is weight function

V is hidden layer

Y is output layer

By using the above network, the mathematical model was calculated as follows to find the output layer.

The input layer is represented by the following matrix

$$X = \begin{Bmatrix} X_1 \\ X_2 \\ . \\ . \\ . \\ X_m \end{Bmatrix}$$

The weight functions are represented by the following matrix

$$W = \begin{Bmatrix} W_{11}^{\,n}, W_{12}^{\,n}, \ldots, & W_{1n}^{\,n} \\ W_{21}^{\,n}, W_{22}^{\,n}, \ldots, & W_{2n}^{\,n} \\ . & . \\ . & . \\ . & . \\ W_{m1}^{\,n}, W_{m2}^{\,n}, \ldots, & W_{mn}^{\,n} \end{Bmatrix}$$

The hidden layer was calculated by the following equation using the above matrices.

$$V = \begin{Bmatrix} W_{11}^{\,n}, W_{12}^{\,n}, \ldots, & W_{1n}^{\,n} \\ W_{21}^{\,n}, W_{22}^{\,n}, \ldots, & W_{2n}^{\,n} \\ . & . \\ . & . \\ . & . \\ W_{m1}^{\,n}, W_{m2}^{\,n}, \ldots, & W_{mn}^{\,n} \end{Bmatrix} * \begin{Bmatrix} X_1 \\ X_2 \\ . \\ . \\ . \\ X_m \end{Bmatrix}$$

$$V_n = (W_{11}^{\,n} * X_1) + (W_{21}^{\,n} * X_2) + \ldots + (W_{mn}^{\,n} * X_m)$$

After determining the hidden layers, the activation function was calculated as shown bellow

$$F(V_1) = 1/ (1+e\text{^-}V_1)$$

$$V_2 = (W_{12}^{1} * X_1) + (W_{22}^{1} * X_2) + (W_{32}^{1} * X_3)$$

$$F(V_2) = 1/ (1+e\text{^-}V2)$$

Likewise, the other hidden layers were calculated until n times to find V_n.

After doing this the output was calculated as follows, however the artificial neural network can have multiple outputs.

$$Y = \begin{Bmatrix} W_{11}^{n+1}, W_{12}^{n+1}, .., & W_{1n}^{n+1} \\ W_{21}^{n+1}, W_{22}^{n+1}, ..., & W_{2n}^{n+1} \\ . & . \\ . & . \\ . & . \\ W_{m1}^{n+1}, W_{m2}^{n+1}, ..., & W_{mn}^{n+1} \end{Bmatrix} * \begin{Bmatrix} V_1 \\ V_2 \\ . \\ . \\ . \\ V_m \end{Bmatrix}$$

$$Y_1 = (W_{11}^{n+1} * V_1) + (W_{21}^{n+1} * V_2) + \ldots + (W_{m1}^{n+1} * V_m)$$

Lastly, the output layer was calculated as shown as $F(Y_1) = 1/ (1+e\text{^-}Y_1)$

As per Back Propagation algorithm the weights will be adjusted automatically for several iterations until I get the desired output.

Back-propagation algorithm defines the strategy of weight adjustment in multilayer neural networks using the gradient optimization method.

The Back-propagation algorithm converges a set of weights that minimizes the mean-square error

$$E = (target - output)^2$$

The weights modification formula:

$$W_{new} = W_{old} + \alpha(Desired - Output) * x_i$$

$W_{11(new)} = W_{11(old)} + \alpha(Desired - Output) * x_1$

$W_{21(new)} = W_{21(old)} + \alpha(Desired - Output) * x_2$

$W_{31(new)} = W_{31(old)} + \alpha(Desired - Output) * x_3$

$W_{12(new)} = W_{12(old)} + \alpha(Desired - Output) * x_1$

$W_{22(new)} = W_{22(old)} + \alpha(Desired - Output) * x_2$

$W_{32(new)} = W_{32(old)} + \alpha(Desired - Output) * x_3$

3.3. Design of hardware

For the preparation of artificial neural network resistor, OP741IC, diode, and multiply circuit are needed. This equipment was organized to prepare of artificial neural network. The first step is simulation with Proteus software. Using this simulated software, the hardware was prepared.

To prepare the hardware first, we deeply study the neural network algorithm and its previous implementations based on Software as well as Hardware. We start by exploring different available techniques. Next, the best approach for our optimization will select by studying different available hardware optimization techniques. Lastly, we combined these and other optimization techniques and come up with the improved design.

The artificial neural network was constructed by using simulation software. The input variables are X_1, X_2 and X_3. The weights are W_1, W_2 and W_3 and they are random vector. Then after the input and the weight will be multiplied after multiplication the output are added by using summing circuit. The summing circuit gives negative voltage by getting the value from summing circuit it enters inverter. The work of inverter is operational amplifier. Using activation function circuit, it is activated and gave as negative value. So it will be inverted to positive by using inverter circuit. Then V_1 and V_2 give the output Y_1. If the value of Y_1 is not the anticipated result it will be back propagated by using back propagation algorithm. Back propagation means adjusting the weight. To adjust the weight there is formula and circuit. On the circuit differential summing and multiply circuit are there.

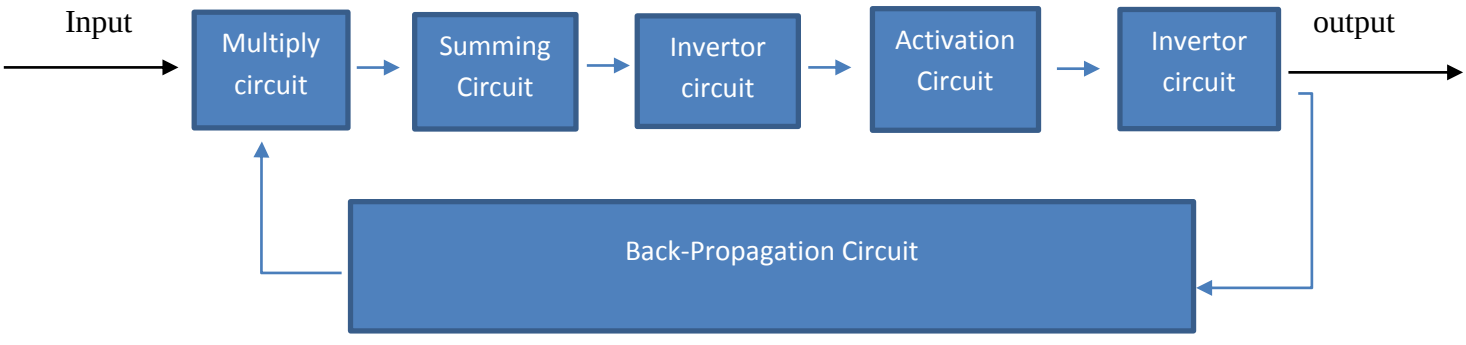


Figure 9 : Block Diagram representation of a neuron network with back-propagation

Chapter 4

Results and Discussions

4.1. Simulation Result

This section describes how to simulate the artificial neural network using proteus version 8.0 software. This software enables to calculate the mathematical model prepared for artificial neural network.

The output of a neuron is a function of the weighted sum of the inputs plus a bias the function of the entire neural network is simply the computation of the outputs of all the neurons. An entirely deterministic calculation Applied to the weighted sum of the inputs of a neuron to produce the output Majority of NN's use sigmoid functions smooth, continuous, and monotonically increasing (derivative is always positive) Bounded range - but never reaches max or min Consider "ON" to be slightly less than the max and "OFF" to be slightly greater than the min.

The most common sigmoid function used is the logistic function.$F(x) = \frac{1}{(1+e^{-x})}$. The calculation of derivatives is important for neural networks and the logistic function has a very nice derivative f'(x) = f(x) (1 - f(x)) other sigmoid functions also used hyperbolic, tangent arctangent. The exact nature of the function has little effect on the abilities of the neural network.[16]

Back propagation algorithm was used in the designing.

- Back-propagation algorithm defines the strategy of weight adjustment in multilayer neural networks using the gradient optimization method.
- The Back-propagation algorithm converges a set of weights that minimizes the mean-square error

$$E = \left(target - output\right)^2$$

- If the output is not correct, the weights were adjusted according to the formula:

Below are the step by step images of simulation.

For the circuit X_1, X_2 and X_3 are the inputs.

When X_1, X_2 and X_3 are given to the circuit with weights W_{11}, W_{21} and W_{31} we get V_1 which is equal to 0.26 V.

Below is the calculation that is the same to the simulation value.

$V_1 = (W_{11}^{(1)} * X_1) + (W_{21}^{(1)} * X_2) + (W_{31}^{(1)} * X_3)$ 4.1

$= (0.2*0.3) + (0.4*0.2) + (0.3*0.4)$

$= 0.26$ V

And when V_1 is given to activation function we get the $F(V_1)$

$F(V_1) = 1/(1+e^{-V_1})$ 4.2

$= 1/(1+e^{-0.26})$

$F(V_1) = 0.5646$ V

From Eq(4.1) & Eq (4.2) Calculations $V_1 = 0.26V$ and $F(V_1) = 0.5646V$

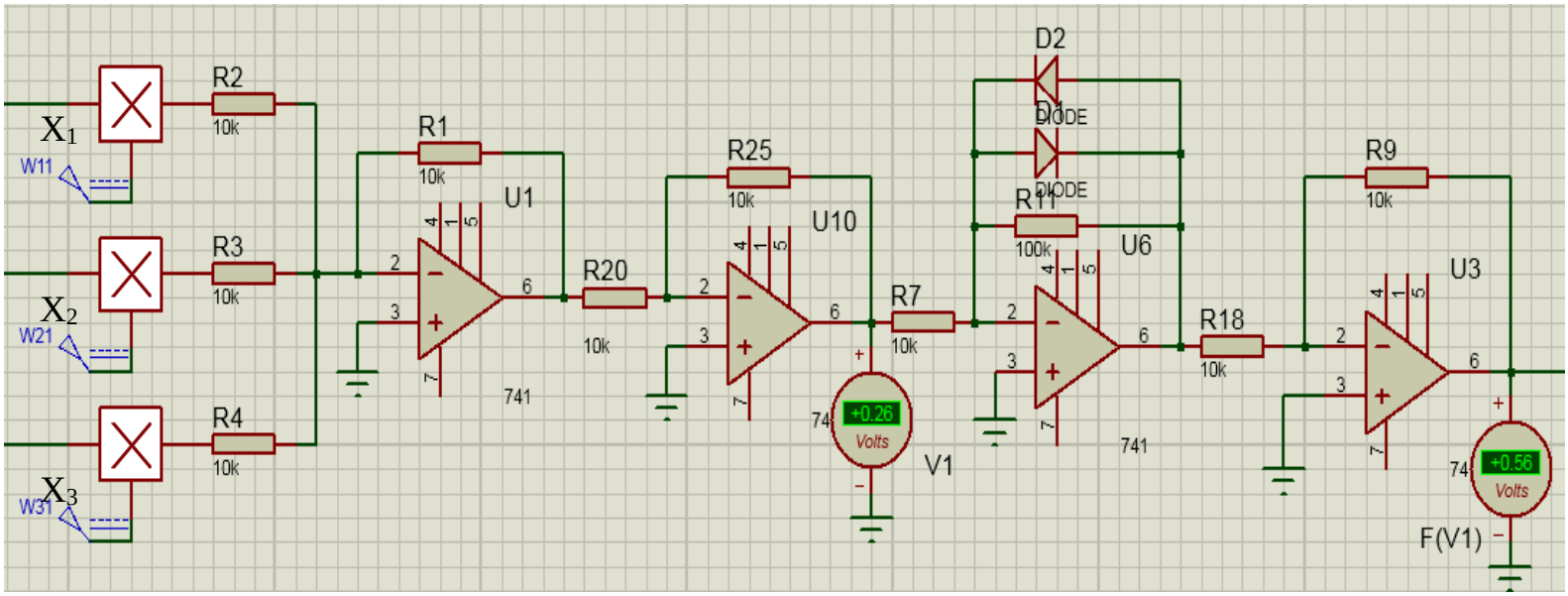


Figure 10 Neural network Hardware categories

When X_1, X_2 and X_3 are given to the circuit with weights W_{12}, W_{22} and W_{32} we get V_2 which is equal to 0.17 V.

$V_2 = (W_{12}^{(1)} * X_1) + (W22^{(1)} * X_2) + (W_{32}^{(1)} * X_3)$ 4.3

$= (0.1*0.3) + (0.3*0.2) + (0.2*0.4)$

$= 0.17$V

When V_2 is given to activation function we get the F (V_2)

$F(V_1) = 1/(1+e^{-V_2})$ ……..4.4

$F(V_2) = 1/(1+e^{-0.17})$

$= 0.5424V$

From Eq(4.3) & Eq (4.4) Calculations | $V_2 = 0.17V$ | and | $F(V_2) = 0.5424V$ |

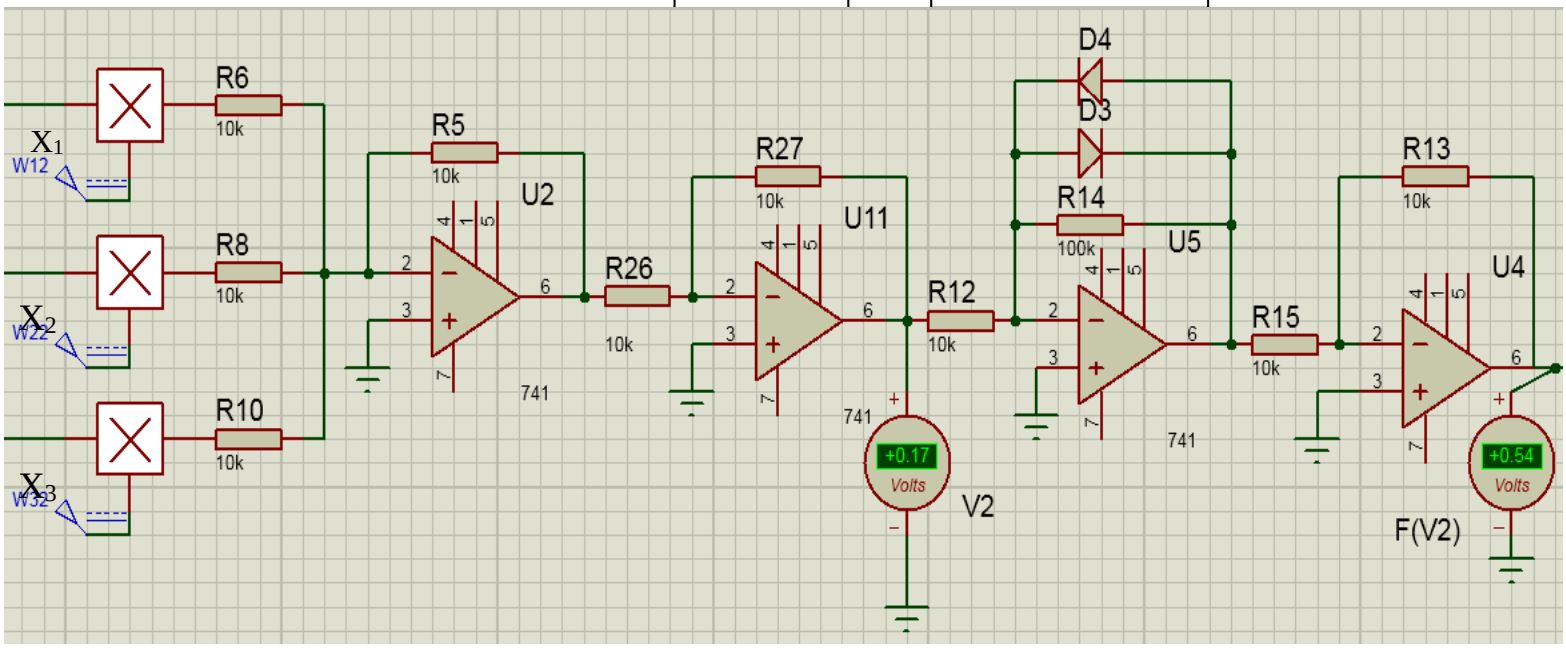


Figure 11 Simulation Result showing V2 voltage and F(V2) voltage.

Now $F(V_1)$ and $F(V_2)$ are applied to adder along with weights $W_{11(1)}$ and $W_{12(1)}$ then we get Y_1 which is equal to 0.38 V.

$Y_1 = (W_{11}^{(2)} * V_1) + (W_{21}^{(2)} * V_2)$ ……..4.5

$= (0.4*0.56) + (0.3*0.54)$

$= 0.386$ V

When Y_1 is given to activation function we get the $F(Y_1)$

$F(Y_1) = 1/(1+e^{-Y_1})$ ……..4.6

$F(Y_1) = 1/(1+e^{-0.386})$

$= 0.596$ V

From Eq(4.5) & Eq (4.6)Calculations $Y_1 = 0.38$ V and $F(Y_1) = 0.596$ V.

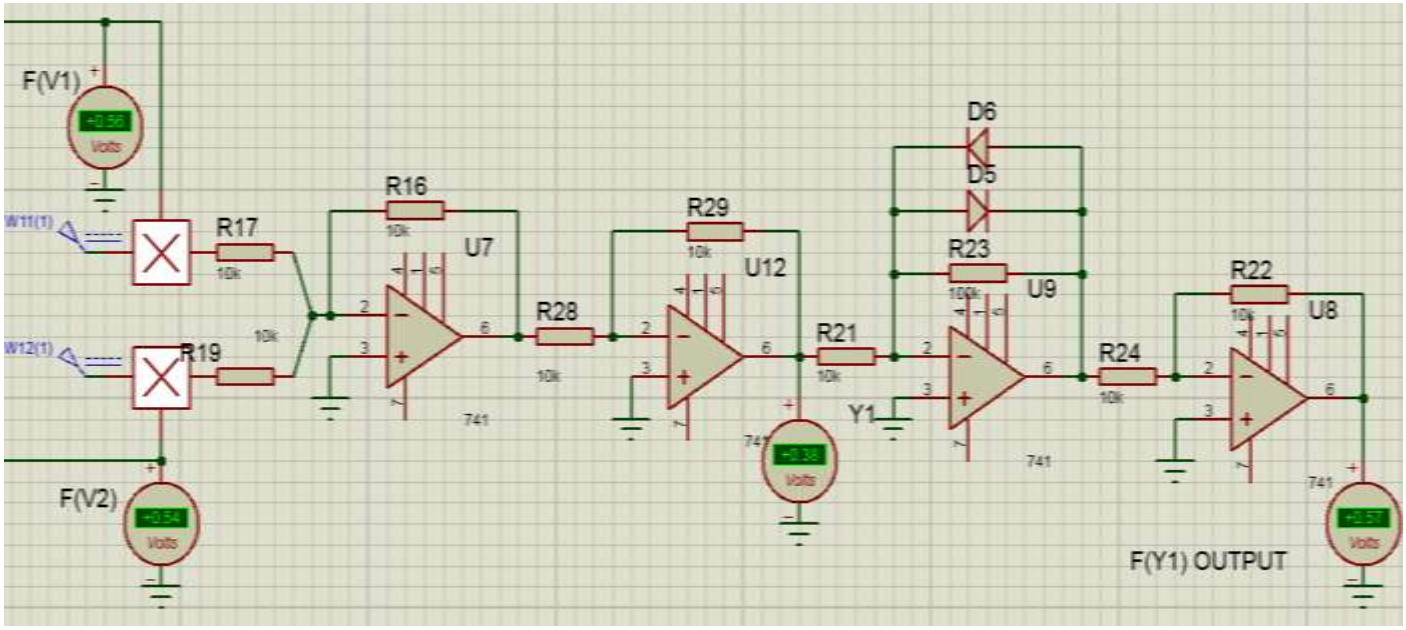


Figure 12 Simulation Result showing Y_1 voltage and Output $F(Y_1)$ voltage.

The complete circuit is shown below

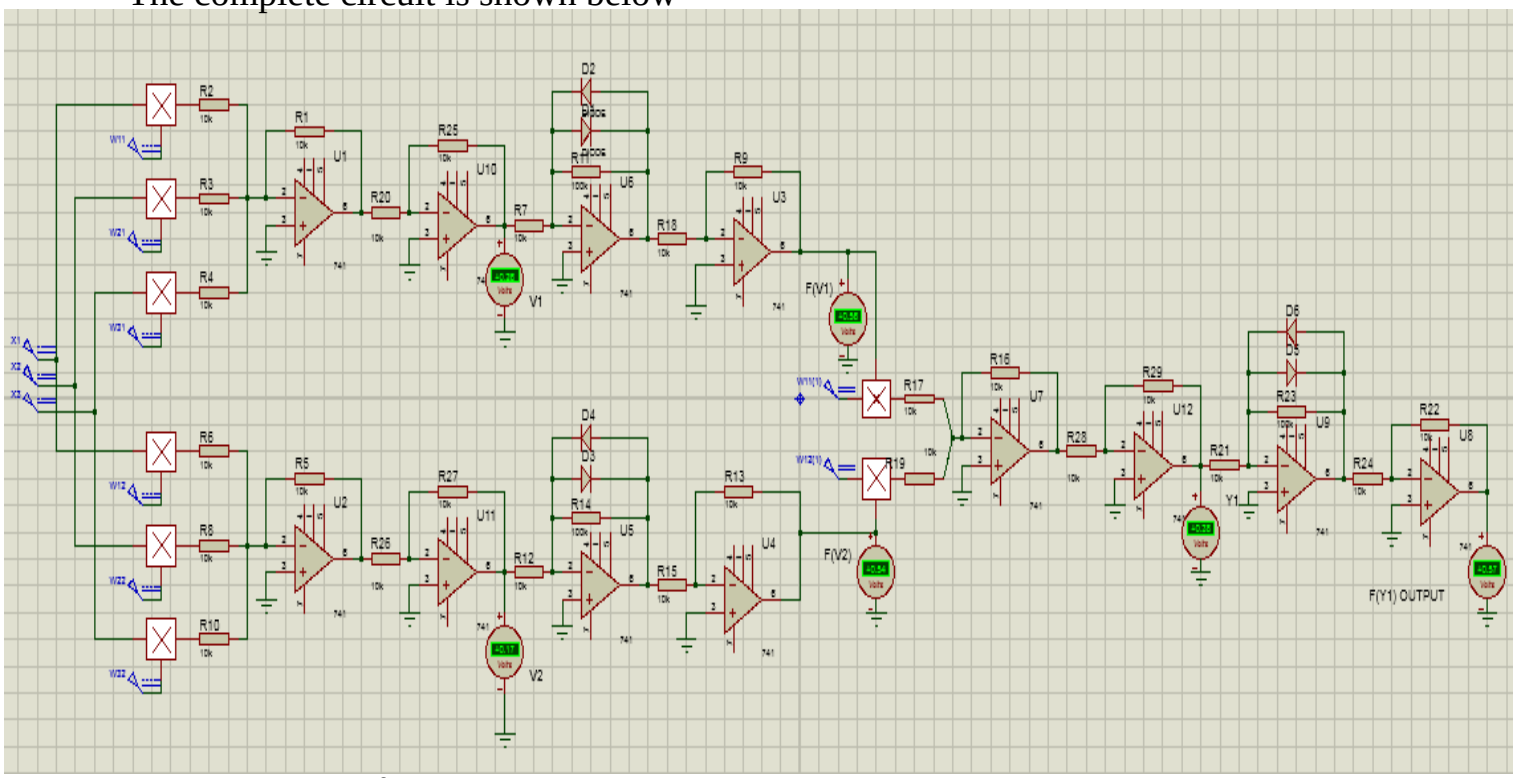


Figure 13 Images of simulation Result using proteus

The output which we got is 0.57V, But the desired output should be 0.65V

To get the desired output 0.65V, I had used Back propagation algorithm.

As per Back Propagation algorithm the weights will be adjusted automatically for several iterations until I get the desired output.

Back-propagation algorithm defines the strategy of weight adjustment in multilayer neural networks using the gradient optimization method. [16]

The Back-propagation algorithm converges a set of weights that minimizes the mean-square error

$$E = \left(target\ -output\right)^2$$

from sample example, Target or Desired value 0.65, Output value 0.596

- $E = \left(0.65\ - 0.596\right)^2$

  = 0.002916 $\longrightarrow$ 0.0005

If the output is not correct, the weights are adjusted according to the formula:

The weights modification formula:

$$W_{new} = W_{old} + \alpha(Desired - Output) * x_i \quad \text{.....4.7}$$

In figure 12 the circuit related to eq(4.7) weights modification formula

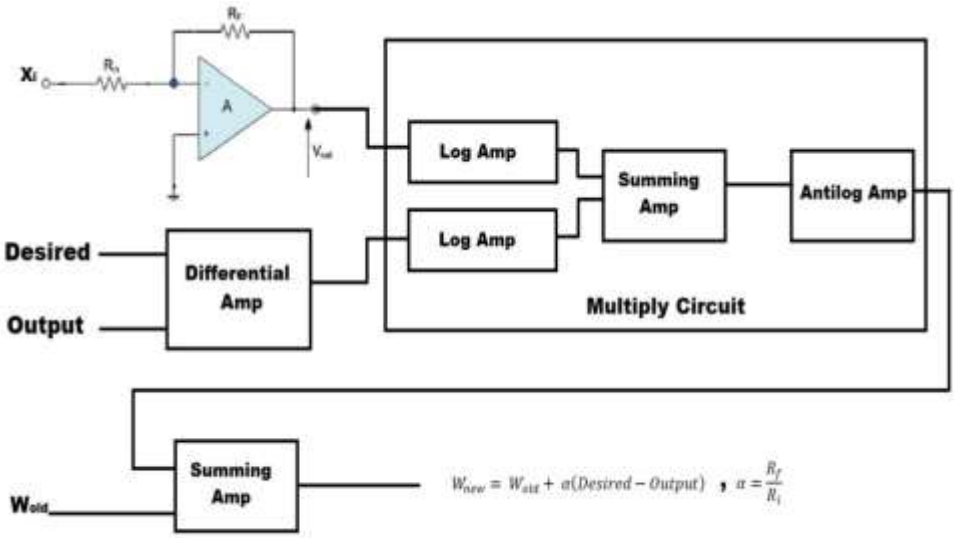


Figure 14 A block diagram of back propagation

Weight modification:

$$W_{new} = W_{old} + \alpha(Desired - Output)X_i$$

For Initial Iteration:

$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.1 \\ 0.4 & 0.3 \\ 0.3 & 0.2 \end{bmatrix} , \quad \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.3 \end{bmatrix} , \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.4 \end{bmatrix}$$

Desired value 0.65V, Output value 0.59V and α=0.6V

For Second Iteration:

$$W_{new} = W_{old} + \alpha(Desired - Output)\mathrm{X}_i$$

$W^{(1)}_{11(new)} = W^{(1)}_{11(old)} + \alpha(Desired - Output) * x_1$ $W^{(1)}_{11(new)} =$ 0.2+0.6*(0.65-0.59) *0.3 =0.21

$W^{(1)}_{21(new)} = W^{(1)}_{21(old)} + \alpha(Desired - Output) * x_2$ $W^{(1)}_{21(new)} =$ 0.4+0.6*(0.65-0.59) *0.2 =0.41

$W^{(1)}_{31(new)} = W^{(1)}_{31(old)} + \alpha(Desired - Output) * x_3$ $W^{(1)}_{31(new)} =$ 0.3+0.6*(0.65-0.59) *0.4 =0.32

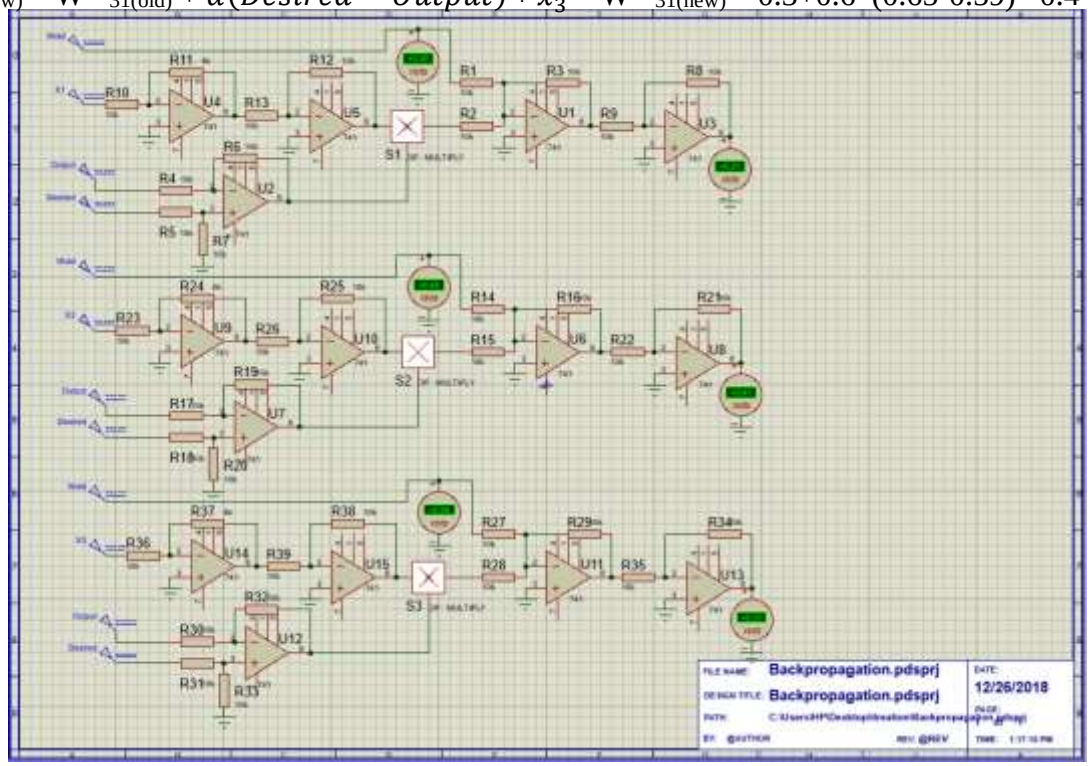


Figure 15 Simulation Result showing a new weight W_{11} –W_{13} voltage.

$W^{(1)}_{12(new)} = W^{(1)}_{12(old)} + \alpha(Desired - Output) * x_1$, $W^{(1)}_{12(new)}$ = 0.1+0.6*(0.65-0.59) *0.3 =0.11

$W^{(1)}_{22(new)} = W^{(1)}_{22(old)} + \alpha(Desired - Output) * x_2$, $W^{(1)}_{22(new)}$ = 0.3+0.6*(0.65-0.59) *0.2 =0.31

$W^{(1)}_{32(new)} = W^{(1)}_{32(old)} + \alpha(Desired - Output) * x_3$, $W^{(1)}_{32(new)}$ = 0.2+0.6*(0.65-0.59) *0.4 =0.21

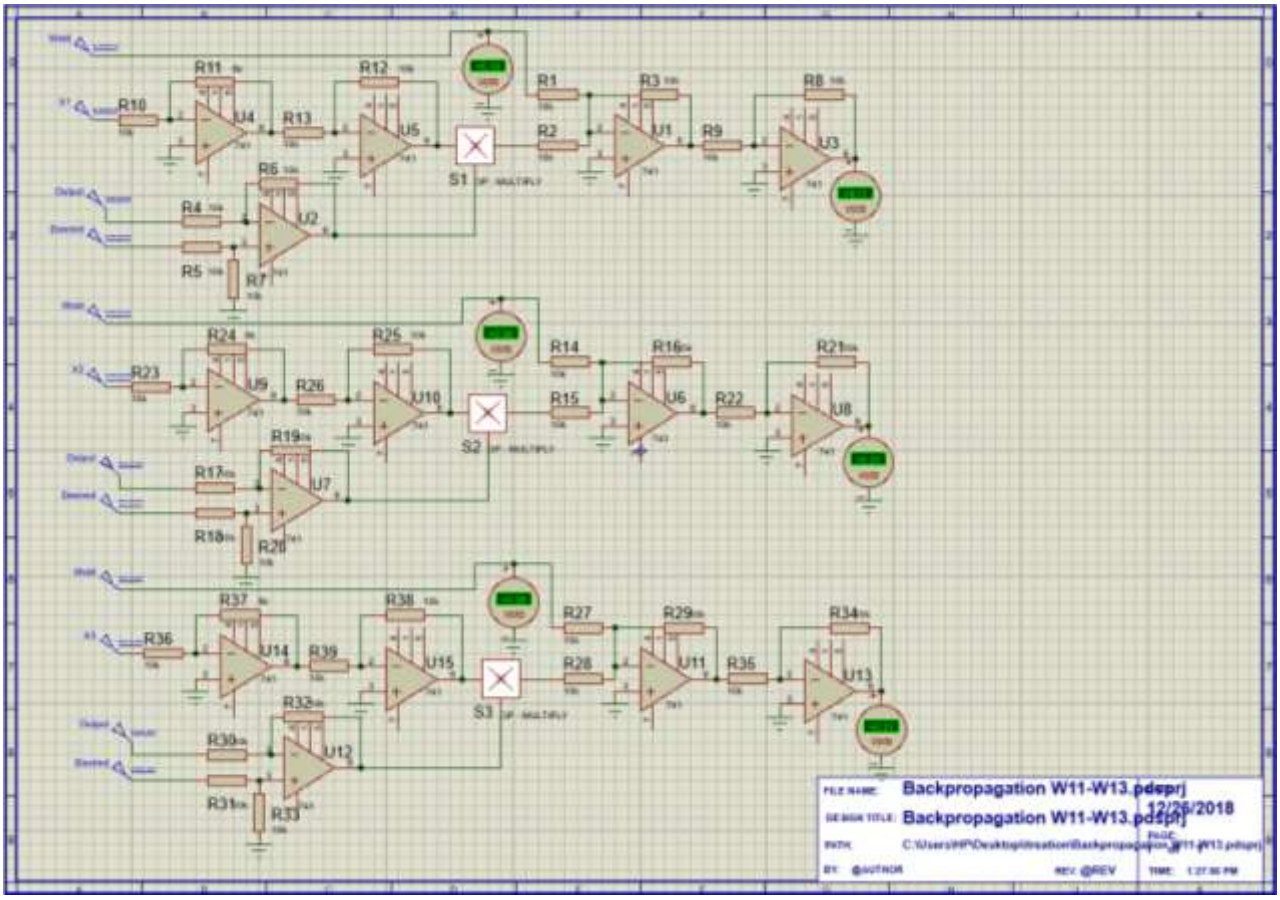


Figure 16 Simulation Result showing a new weight W_{21} –W_{23} voltage.

$W^{(2)}_{11(new)} = W^{(2)}_{11(old)} + \alpha(Desired - Output) * v_1$, $W^{(2)}_{11(new)}$ =0.4+0.6*(0.65-0.57) *0.56 =0.43

$W^{(2)}_{21(new)} = W^{(2)}_{21(old)} + \alpha(Desired - Output) * v_1$, $W^{(2)}_{11(new)}$ =0.3+0.6*(0.65-0.57) *0.54 =0.33

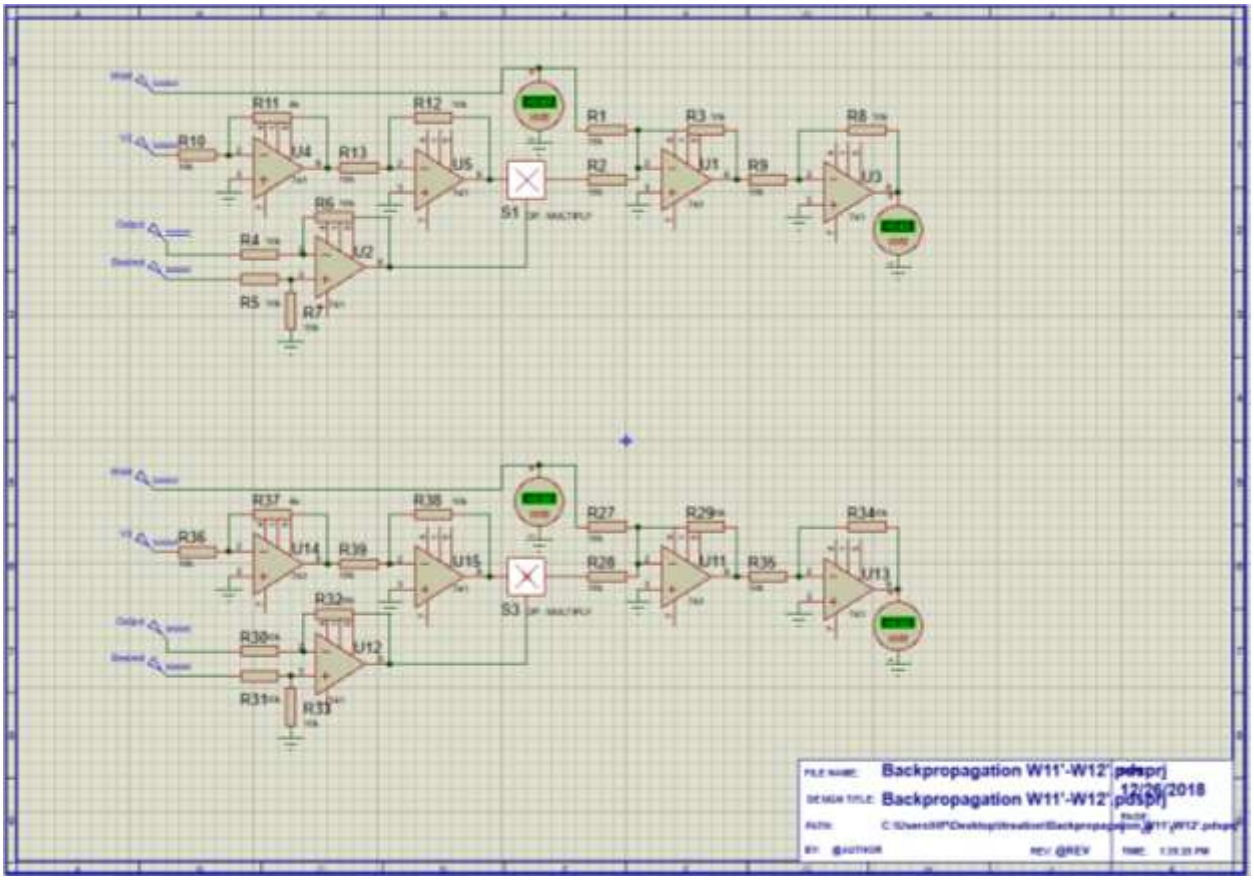


Figure 17 Simulation Result showing a new weight W^1_{11} –W^1_{12} voltage.

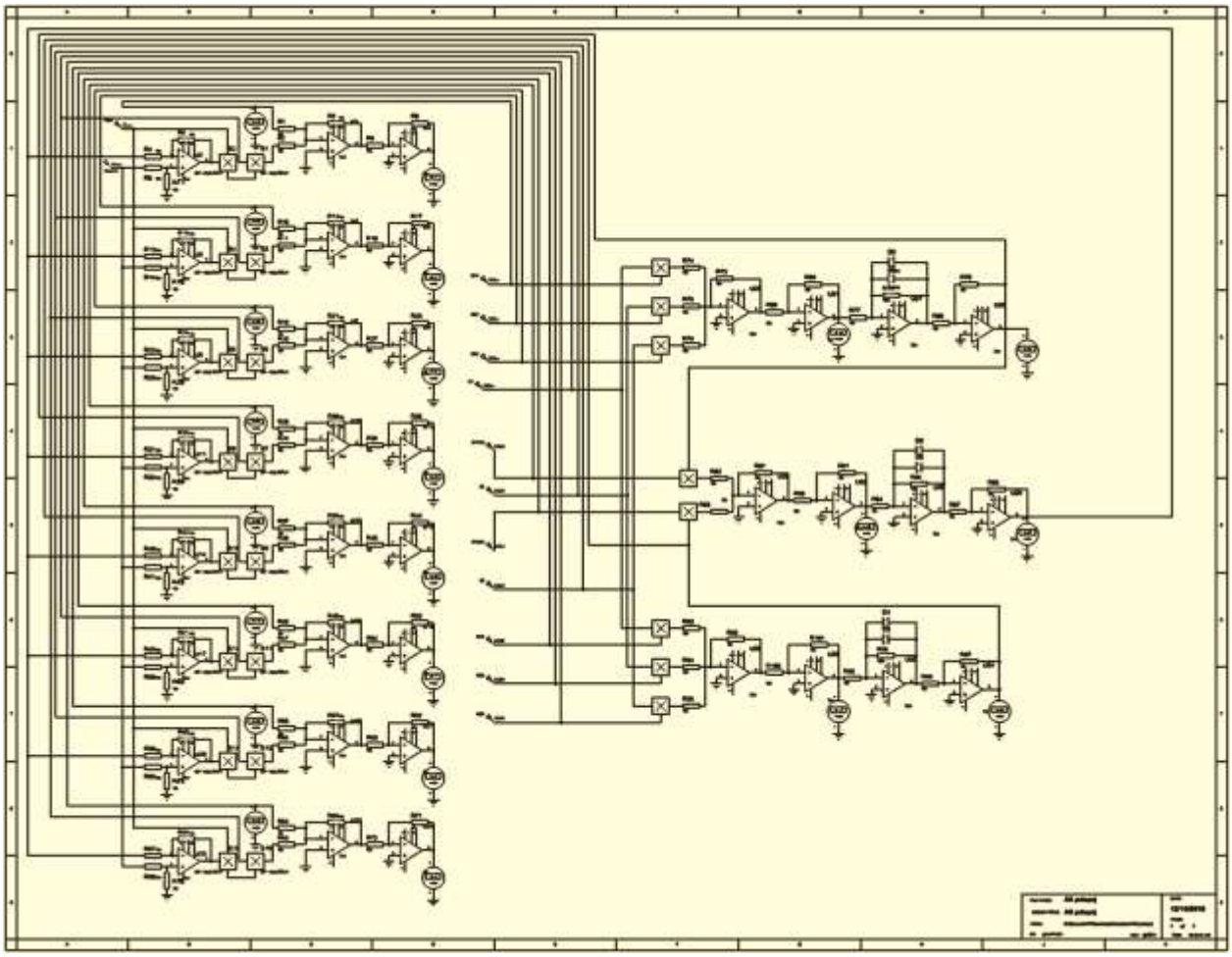


Figure 18 Simulation Result showing a neural network with back propagation

For 1st Iteration

$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} \end{bmatrix} = \begin{bmatrix} 0.21 & 0.11 \\ 0.41 & 0.31 \\ 0.31 & 0.21 \end{bmatrix} \quad , \quad \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \end{bmatrix} = \begin{bmatrix} 0.42 \\ 0.32 \end{bmatrix} , \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.4 \end{bmatrix}$$

For the circuit X_1, X_2 and X_3 are the inputs.

When X_1, X_2 and X_3 are given to the circuit with weights W_{11}, W_{21} and W_{31} we get V_1 which is equal to 0.26 V.

From eq(4.1) is the calculation that is the same to the simulation value.

$V_1 = (W_{11}^{(1)} * X_1) + (W_{21}^{(1)} * X_2) + (W_{31}^{(1)} * X_3)$

$= (0.21*0.3) + (0.41*0.2) + (0.31*0.4)$

$= 0.27$ V

And when V_1 is given to activation function we get the $F(V_1)$

$F(V_1) = 1/ (1+e^{\wedge}-0.27)$

$F(V_1) = 0.567$ V

From eq(4.2) Calculations $V_1 = 0.27$V and $F(V_1) = 0.567$V

When X_1, X_2 and X_3 are given to the circuit with weights W_{12}, W_{22} and W_{32} we get V_2 which is equal to 0.17 V.

$V_2 = (W_{12}^{(1)} * X_1) + (W_{22}^{(1)} * X_2) + (W_{32}^{(1)} * X_3)$

$= (0.11*0.3) + (0.31*0.2) + (0.21*0.4)$

$= 0.18$V

When V_2 is given to activation function we get the $F(V_2)$

$F(V_2) = 1/ (1+e^{\wedge}-0.18)$

$= 0.544$V

From eq (4.3) Calculations $V_2 = 0.18$V and $F(V_2) = 0.544$V

Now $F(V_1)$ and $F(V_2)$ are applied to adder along with weights $W_{11\,(1)}$ and $W_{12\,(1)}$ then we get Y_1 which is equal to 0.42 V.

$Y_1 = (W_{11}^{(2)} * V_1) + (W_{21}^{(2)} * V_2)$

$= (0.42*0.567) + (0.32*0.544)$

$= 0.41$ V

When Y_1 is given to activation function we get the $F(Y_1)$

$F(Y_1) = 1/ (1+e^{\wedge}-0.42)$

$= 0.60$ V

From eq (4.4) Calculations $Y_1 = 0.41$ V and $F(Y_1) = 0.60$ V.

Desired value 0.65V, Output value 0.60105V and α=0.6

Error $E = (0.65 - 0.60105)^2$

$= 0.002396 \longrightarrow 0.0005$V

The First iteration neuron output is 0.596,

The second iteration neuron output is 0.60105

Target value is 0.65 V

Like this continuously several iterations are repeated until we get the desired value. Below table shows the different iteration results.

Table 1: Result for mathematical and simulation model

Iteration	Mathematical Model		Simulation Model	
	Output	Error	Output	Error
Initial	0.596	0.00292	0.591	0.00348
1st	0.601	0.0024	0.602	0.0023
2nd	0.603	0.00221	0.609	0.00168
3rd	0.605	0.00203	0.614	0.00129
4th	0.607	0.00185	0.618	0.001024
5th	0.609	0.00168	0.621	0.00084
6th	0.611	0.00152	0.623	0.00072
7th	0.613	0.00137	0.625	0.000625
8th	0.615	0.00123	0.627	0.000529
9th	0.617	0.00109	0.629	0.000441
10th	0.619	0.00096	0.630	0.0004
11th	0.621	0.00084	0.631	0.000361
12th	0.623	0.00073	0.633	0.000289
13th	0.625	0.00063	0.634	0.000256

14th	0.627	0.00053	0.635	0.000225
15th	0.629	0.00044	0.636	0.000196
16th	0.631	0.00036	0.637	0.000169
17th	0.633	0.00029	0.637	0.000169
18th	0.635	0.00023	0.638	0.000144
19th	0.637	0.00017	0.639	0.000121
20th	0.639	0.00012	0.639	0.000121
21th	0.641	0.000081	0.640	0.0001
22th	0.643	0.000049	0.641	0.000081
23th	0.645	0.000025	0.641	0.000081
24th	0.647	0.000009	0.642	0.000064
25th	0.649	0.000001	0.643	0.000049
26th			0.643	0.000049
27th			0.644	0.000036
28th			0.644	0.000036
29th			0.645	0.000025
30th			0.646	0.000016
31th			0.646	0.000016
32th			0.647	0.000009
33th			0.647	0.000009
34th			0.648	0.000004
35th			0.649	0.000001
36th			0.650	0

So after 25 iterations the error is minimized and we get the desired output.

4.2. Discussion:

The artificial neural network was designed by using op-amp, multiply, summing, differential, and activation function circuit. These circuits by combining together give the artificial neural networks which have the same characteristic like biological neural network. The model which was designed by using simulation software shows a correct output just like mathematical. Hence it can be said that this it is a hardware implementation of Soft Neural network.

During testing by inserting constant input values for X_1, X_2 and X_3 and $W_{11}^{(1)}$, $W_{12}^{(1)}$, $W_{13}^{(1)}$, $W_{21}^{(1)}$, $W_{22}^{(1)}$, $W_{23}^{(1)}$ which are found on the input layer and $W_{11}^{(2)}$, $W_{12}^{(2)}$, which are found in the hidden layer the output (Y), is checked. However, the expected results and output values have difference. To solve this problem back propagation was used. Back Propagation is implemented in order to adjust the weights for getting desired output. So, to get the desired output around 25 iterations were done for mathematical model. After 25 iterations the output is equal to the desired value and error is eliminated completely. But in the simulation the result approached the mathematical model after 36 iterations the o/p is equal to the desired value. The variation in the iteration is not a big deal because it was expected to get the value by circuit. The main concern is showing that the circuit learns the change in the values.

The problem here is the weights have to adjust manually for all those iterations. This due to the reason that the circuit for automatic is difficult. However, it can be built by detail investigation to convert manual adjust to automatic.

Chapter Five

Conclusion and Recommendation

5.1. Conclusion

HNN research and applications have witnessed a slow and incremental progress in last two decades. Even though ANN hardware has been there for more than last two decades, the rapid growth in general purpose hardware (microprocessors, DSPs, etc.) did not let most of these implementations to outperform to the extent of becoming commercially successful.

NN software theory with different NN structure and algorithm are commonly available. On the other hand, the hardware base NN could be better studied.

In this thesis general concepts of the different structures of NN dynamics have been studied. Back-Propagation algorithms and we realized the soft NN structure and the dynamic learning algorithms with hardware design and we succeeded to show the off line.

Analog circuit elements could represent neural network elements could represent neural network elements. Activation function, multiplication addition is replaced by analog counter based parts to function as neurons have been shown to have similar numerical values in terms node, weight and output.

The analytical and circuit based neurons have also compare in terms back- propagation iteration values. The analytical was up to 25^{th} iterations to achieve desired value and the analog circuit neuron also achieved the desired value in 36^{th} iteration.

5.2. Recommendation

The author would like to recommend the following points

- Further researchers should be done on artificial neural network.
- Research should be done on automatic switching devices.
- Scholars who are working on this area have to be encouraged.

Reference

1. McCulloch, W. S. and Pitts, W., 1943, A Logical Calculus of the Ideas Immanet in Nervous Activity.Bulletin of Mathematical Biophysics, vol. 5, 115-133, 1943.)
2. Lindsey, C. S., Lindblad, Th., Sekniaidze, G., Minerskjold, M., Szekely, S., and Eide, A., 1995, Experience with the IBM ZISC Neural Network Chip. Proceedings of 3rd Int. Workshop on Software Engineering, Artificial Intelligence, and Expert Systems, for High Energy and Nuclear Physics, Pisa, Italy, April 3-8, 1995.
3. Alspector, J., "VLSI Architecture for Neural Networks", Neural Networks: concepts, applications and Implementations, Vol. 1,pp.180-213, 1991.
4. M. Tanaka and T. Saito, "Neural Nets and Circuits," *CORONA PUBLISHING CO., LTD.*, pp.195–209, 1999.
5. Hammerstrom D., "An Implementation of Kohonen's SelfOrganizing Map on the adaptive solutions neurocomputer", Artificial Neural Networks, pp. 715-720, Elsevier Science (North-Holland).
6. P. Moerland and E. Fiesler, "Neural network adaptations to hardware implementations," *Handbook of Neural Computation*, vol. 1, p. 2, 1997.
7. Valeriu Beiu, " How to Build VLSI-Efficient Neural Chips", Proceedings of the Interantional ICSC Symp. On Engineering of Intelligent System EIS'98, pp. 66-75.
8. Hammerstrom, D., "Neurocomputing Hardware : Present and Future", Artificial Intelligence Review, Vol. 7, 1993, pp 285- 300.
9. H. Hikawa, "*{*FPGA*}* implementation of self-organizing map with digital phase locked loops," *Neural Networks*, vol. 18, no. 56, pp. 514 – 522, 2005, *{*IJCNN*}* 2005. [Online]. Available:http://www.sciencedirect.com/science/article/pii/S0893608005001103
10. Jihan Zhu and Peter Sutton, " FPGA Implementation of Neural Networks – a Survey of a Decade of Progress" Proceedings of 13th International Conference on Field Programmable Logic and Applications (FPL 2003), Lisbon, Sep 2003
11. Ramacher, U., et. Al., "SYNAPSE 1 – A General Purpose Neurocomputer", Siements AG Proprietary Information Manual, February, 1994
12. P. tikovic and D. Durackova, " A Simple Leaky Integrate-andFire Neuron for VLSI Implementation" P Proceedings of the DDECS ' 2001 workshop, Györ, Hungary, Apr. 18 - 20, 2001, pp. 107-11001, pp. 107-110

13. P. Tikovie and D. Durackova, " An Integrate-and-Fire Neuron Implementaion", Electronic Devices and Systems Y2K - Proceedings, Brno, 2000, pp. 93-98
14. Isik Aybay, Semih Cetinkaya and Ugur Halici, " Classification of Neural Netowork Hardware", Neural Network World, IDG Co., Vol. 6 No.1, 1996, pp 11-29.
15. Silvio P. Eberhardt, Raoul Tawel, Timothy X Brown, Taher Daud and AnilKumar P. Thakoor, "Analog VLSI Neural Networks: Implementation Issues and Examples in Optimzation and Supervised Learning", IEEE, 1992
16. R. C. Frye, E. A. Rietman, and C. C. Wong, "Back-propagation learning and nonidealities in analog neural network hardware," *Neural Networks,IEEE Transactions on*, vol. 2, no. 1, pp. 110–117, 1991
17. Murray, A.F., et.al, "Analog Neural VLSI: Issues, Trends and Pulses", Artificial Neural Networks, 1992, pp 35-43, Elsevier Science (North-Holland)
18. H. Djahanshahi, M. Ahmadi, G.A. Jullien and W.C. Miller, "A Self-scaling Neural Hardware Structure that reduces the effect of some Implementation Errors" Proceedings of the 1997 IEEE Workshop, 24-26 Swpt.1997
19. Artificial Neural Networks: a Review of Commercial Hardware Fernando Morgado Diasa, Ana Antunesa, Alexandre Manuel Mota, Escola Superior de Tecnologia de Setúbal, Departamento de Engenharia Electrotécnica, Campus do IPS, Estefanilha, 2914-508 Setúbal, Portugal Proposed in 9 October 2003
20. S. Jung and S. S. Kim, "Hardware implementation of a real-time neural network controller with a dsp and an fpga for nonlinear systems," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 1, pp. 265–271, 2007
21. Sackinger E., et. al, "Hardware Requirements for Neural Network Pattern Classifier", IEEE Micro, February 1992, pp 32-39.
22. Peery Moerlan and Emile Fiesler, "Neural Network Adaptations to Hardware Implementations", January 1997, Handbook of Neural Computation, E1.2:1-13. Institute of Physics, Oxford University Publishing, New York
23. V. Calayir, T. Jackson, A. Tazzoli, G. Piazza, and L. Pileggi, "Neurocomputing and associative memories based on ovenized aluminum nitride resonators," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–8..