OPTICAL MULTISTAGE INTERCONNECTION NETWORKS USING NEURAL NETWORK APPROACH

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Computer Science & Engineering

By:
Kiranpreet Kaur

(Regn. No. 80732010)

Under the supervision of:

Rinkle Aggarwal

Sr. Lecturer, CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

MAY 2009

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"Optical Multistage Interconnection Networks using Neural Network Approach"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Rinkle Aggarwal and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(KIRANPREET KAUR)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Rinkle 04/06/09

(RINKLE AGGARWAL)
Lecturer (SS),
Computer Science and Engineering Department,
Thapar University, Patiala.

Countersigned by

Seema Bawa 04/06/2009

(SEEMA BAWA)
Professor & Head,
Computer Science & Engineering Department,
Thapar University,
Patiala.

RK Sharma 5/6/09

(R.K.SHARMA)
Dean (Academic Affairs),
Thapar University,
Patiala.

ACKNOWLEDGEMENT

The real spirit of achieving a goal is through the way of excellence and austerous discipline. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various personalities.

First of all, I render my gratitude to the ALMIGHTY who bestowed self-confidence, ability and strength in me to complete this work. Without his grace this would never come to be today's reality.

With deep sense of gratitude I express my sincere thanks to my esteemed and worthy Supervisor **Ms. Rinkle Aggarwal (Sr. Lecturer),** Department of Computer Science and Engineering for her valuable guidance in carrying out this work under his effective supervision, encouragement, enlightenment and cooperation. Most of the novel ideas and solutions found in this thesis are the result of our numerous stimulating discussions. Her feedback and editorial comments were also invaluable for writing of this thesis.

I shall be failing in my duties if I do not express my deep sense of gratitude towards **Dr. Seema Bawa,** Professor and Head of Computer Science and Engineering Department who has been a constant source of inspiration for me throughout this work.

I am greatful to **Dr. R.K. Sharma**, Dean of Academic Affair for his constant encouragement that was of great importance in the completion of the thesis.

I am also thankful to all the staff members of the Department for their full cooperation and help.

My greatest thanks are to all who wished me success especially my parents, my sisters whose support and care makes me stay on earth.

Place: TU, Patiala

Date: 2-6-2009

(KIRANPREET KAUR)

ABSTRACT

Now a days Optical Multistage Interconnection Networks (OMINs) are being used as communication media for distributed computing systems. Neural network solution is used in case of OMINs in order to avoid crosstalk. In this thesis neural network routing methodologies are discussed that can be used to generate control bits for a broad range of OMINs. The routing methodology makes use of an Artificial Neural Network (ANN) that functions as a parallel computer for generating the routes. The parallel nature of the ANN computation may make this routing scheme faster than conventional routing scheme. A neural network computation algorithm is used to solve for the optimal traffic routing in a general N-node communication network. It has been observed that neural network gives better results in terms of speed and avoid crosstalk.

Artificial neural networks have been studied for many years in the hope of achieving human like performance in the fields of speech and object recognition. These networks composed of many nonlinear computational elements operating in parallel and arranged in patterns reminiscent of biological neural nets. The routing methodology makes use of an ANN that functions as a parallel computer for generating the routes. Neural Networks, which are simplified models of the biological neuron system, is a massively parallel distributed processing system made up of highly interconnected neural computing elements that have the ability to learn and thereby acquire knowledge and make it available for use.

Neural Networks are a class of systems that have many simple processors-neurons-which are highly interconnected. A neural network is a special form of parallel computer as the function of each neuron is simple, and the behavior is determined predominately by the set of interconnection. Neural Networks have been used to solve a wide range of problems. In this thesis a set of methods have been discussed for developing neural networks that can be used to systematically and repeatably engineer neural networks to solve specific problems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 INTRODUCTION

1. Neural Network

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 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. An Artificial Neural Network is a network of many very simple processors units, each possibly having a small amount of local memory. The units are connected by unidirectional communication channels connections, which carry numeric as opposed to symbolic data. The units operate only on their local data and on the inputs they receive via the connections.

The design motivation is what distinguishes neural networks from other mathematical techniques. A neural network is a processing device, either an algorithm, or actual hardware, whose design was motivated by the design and functioning of human brains and components thereof.

There are many different types of Neural Networks, each of which has different strengths particular to their applications. The abilities of different networks can be related to their structure, dynamics and learning methods.

Neural Networks offer improved performance over conventional technologies in areas which includes: Machine Vision, Robust Pattern Detection**,** Signal Filtering**,** Virtual

Reality, Data Segmentation, Data Compression, Data Mining, Text Mining, Artificial Life, Adaptive Control, Optimisation and Scheduling, Complex Mapping and more.

1.1 Biological Neuron

The brain is principally composed of about 10 billion neurons, each connected to about 10,000 other neurons. Each of the yellow blobs in the picture above are neuronal cell bodies (soma), and the lines are the input and output channels (dendrites and axons) which connect them.

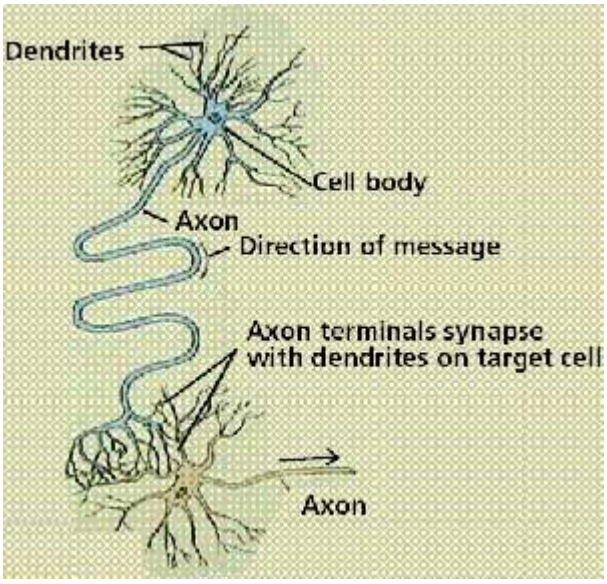


Figure 1. Biological Neuron[15]

Each neuron receives electrochemical inputs from other neurons at the dendrites. If the sum of these electrical inputs is sufficiently powerful to activate the neuron, it transmits an electrochemical signal along the axon, and passes this signal to the other neurons whose dendrites are attached at any of the axon terminals. These attached neurons may then fire. It is important to note that a neuron fires only if the total signal received at the cell body exceeds a certain level. The neuron either fires or it does not, there are not different grades of firing.

So, our entire brain is composed of these interconnected electro-chemical transmitting neurons. From a very large number of extremely simple processing units, each performing a weighted sum of its inputs, and then firing a binary signal if the total input exceeds a certain level, the brain manages to perform extremely complex tasks.

This is the model on which artificial neural networks are based. Thus far, artificial neural networks have not even come close to modeling the complexity of the brain, but they have shown to be good at problems which are easy for a human but difficult for a traditional computer, such as image recognition and predictions based on past knowledge.

How the Human Brain Learns?

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

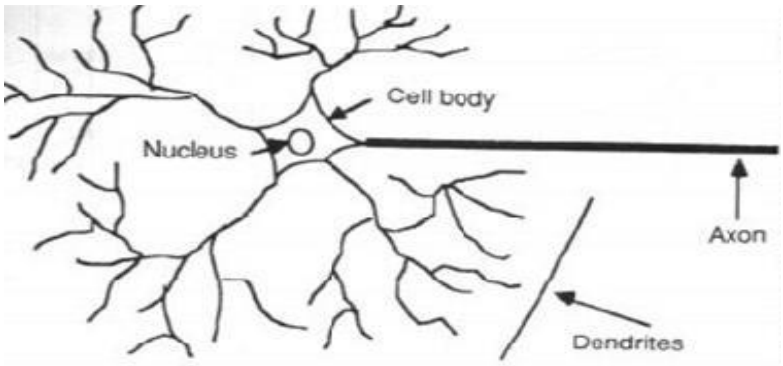


Figure 2. Components of a neuron[15]

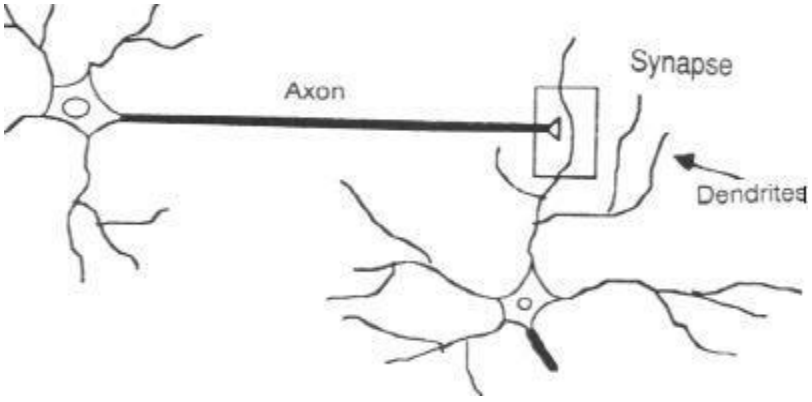


Figure 3. Synapse[15]

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 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[15].

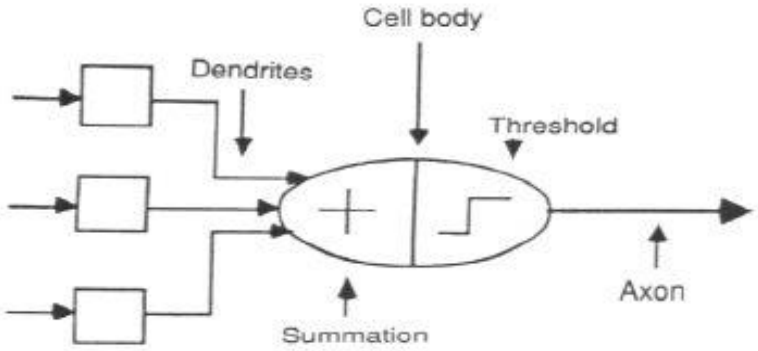


Figure 4. The Neuron Model[15]

Artificial neural networks have been studied for many years in the hope of achieving human – like performance in the fields of speech and image recognition. These networks composed of many nonlinear computational elements operating in parallel and arranged in patterns reminiscent of biological neural nets. This thesis presents a study about the use of neural network computational algorithms for optical networks. These algorithms are used to determine optimal traffic routing for communication networks. Neural network solution is used in case of Optical Multistage Interconnection Networks(OMINs) in order to avoid crosstalk. The routing methodology makes use of an ANN that functions as a parallel computer for generating the routes. It has been observed that neural network gives better results in terms of speed and crosstalk avoidance. Neural Networks(NNs) are simplified imitations of the central nervous system, and obviously therefore, have been motivated by the kind of computing performed by the human brain. The structural constituents of a human brain termed neurons are the entities, which perform computations such as cognition, logical inference, pattern recognition and so on. Hence the technology, which has been built on a simplified imitation of computing by neurons of a brain, has been termed Artificial Neural Systems(ANS) technology or Artificial Neural Networks(ANN) or simply Neural Networks. A human brain develops with time and this, in common parlance is known as experience. An artificial neuron is a device with many inputs and one output.

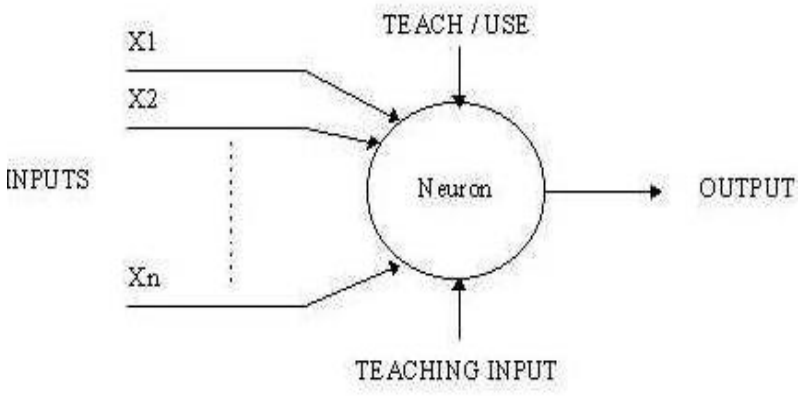


Figure 5. An artificial neuron with n inputs and one output[21]

The stability-plasticity issue is of great importance to NN architectures. The NN needs to remain plastic to significant or useful information but remain stable when presented with irrelevant information.

1.2 A More Complicated Neuron

The previous neuron does not do anything that conventional computers do already. A more sophisticated neuron (figure 6) is the McCulloch and Pitts model (MCP). The difference from the previous model is that the inputs are weighted, the effect that each input has at decision making is dependent on the weight of the particular input. The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire.

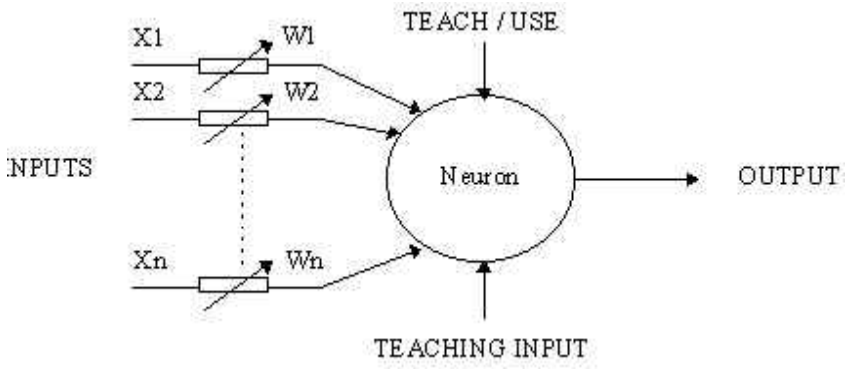


Figure 6. A Complicated Neuron[21]

In mathematical terms, the neuron fires if and only if;

$$X1W1 + X2W2 + X3W3 + ... > T$$

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The MCP neuron has the ability to adapt to a particular situation by

changing its weights and/or threshold. Various algorithms exist that cause the neuron to 'adapt' the most used ones are the Delta rule and the back error propagation. The former is used in feed-forward networks and the latter in feedback networks.

1.3 Historical Background

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several areas. Many important advances have been boosted by the use of inexpensive computer emulations. Following an initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, important advances were made by relatively few researchers. These pioneers were able to develop convincing technology which surpassed the limitations identified by Minsky and Papert. Minsky and Papert, published a book in 1969 in which they summed up a general feeling of frustration against neural networks among researchers, and was thus accepted by most without further analysis. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding.

1.4 Learning methods

1.4.1 Supervised Learning[15]

Supervised learning which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning. An important issue conserning supervised learning is the problem of error convergence, i.e. the minimisation of error between the desired and computed unit values. The aim is to determine a set of weights which minimises the error. One well-known method, which is common to many learning paradigms is the Least Mean Square (LMS) convergence.

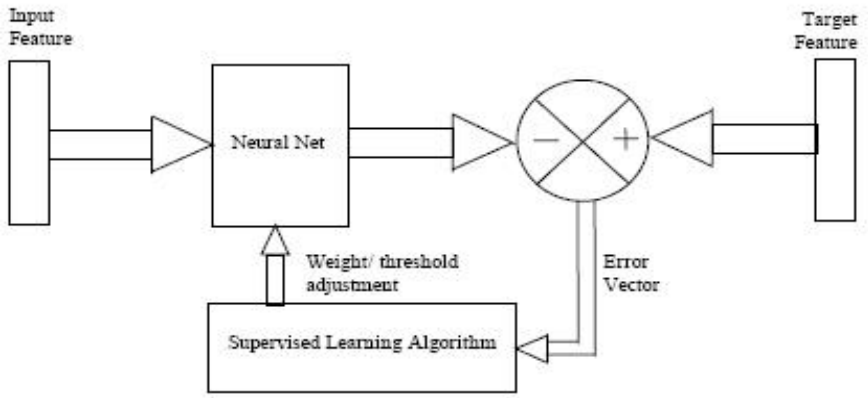


Figure 7. Supervised Learning[15]

1.4.2 Unsupervised Learning[15,21]

Unsupervised Learning uses no external teacher and is based upon only local information. It is also referred to as self-organisation, in the sense that it self-organises data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning. From Human Neurons to Artificial Neuronesther aspect of learning concerns the distinction or not of a seperate phase, during which the network is trained, and a subsequent operation phase. A neural network learns off-line if the learning phase and the operation phase are distinct. A neural network learns on-line if it learns and operates at the same time. Usually, supervised learning is performed off-line, whereas unsupervised learning is performed on-line.

1.4.3 Reinforcement Learning[21]

This type of learning may be considered as an intermediate form of the above two types of learning. Here the learning machine does some action on the environment and gets a

feedback response from the environment. The learning system grades its action good or bad based on the environmental response and accordingly adjusts its parameters. Generally, parameter adjustment is continued until an equilibrium state occurs, following which there will be no more changes in its parameters. The selforganizing neural learning may be categorized under this type of learning.

1.5 Uses

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an expert in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer what if questions.

Other advantages include:

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
- Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

1.6 Applications

1.6.1 Neural Networks in Practice

Given this description of neural networks and how they work, what real world applications are they suited for? Neural networks have broad applicability to real world business problems. In fact, they have already been successfully applied in many industries. Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

- sales forecasting
- industrial process control
- customer research
- data validation
- risk management
- target marketing

But to give some more specific examples. ANN are also used in the following specific paradigms : recognition of speakers in communications, diagnosis of hepatitis, recovery of telecommunications from faulty software, three-dimensional object recognition, hand-written word recognition, and facial recognition.

1.6.2 Neural networks in Medicine

ANNs are currently a hot research area in medicine and it is believed that they will receive extensive application to biomedical systems in the next few years. The examples need to be selected very carefully if the system is to perform reliably and efficiently.

1.6.2.1 Modelling 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.

1.6.2.2 Electronic Noses

ANNs are used experimentally to implement electronic noses. Electronic noses have several potential applications in telemedicine. Telemedicine is the practice of medicine over long distances via a communication link.

1.7 Neural Network Topologies

1.7.1 Feed-forward Neural Networks

Feed-forward neural networks, where the data from input to output units is strictly feedforward. The data processing can extend over multiple layers of units, but no feedback connections are present, that is, connections extending from outputs of units to inputs of units in the same layer or previous layers. Classical examples of feed-forward neural networks are: the Perceptron and Adaline.

1.7.2 Recurrent Neural Networks

Recurrent neural networks that do contain feedback connections. Contrary to feed-forward networks, the dynamical properties of the network are important. In some cases, the activation values of the units undergo a relaxation process such that the neural network will evolve to stable state in which these activations do not change anymore. In other applications, the change of the activation values of the output neurons are significant, such that the dynamical behaviour constitutes the output of the neural network.

1.8 Advantages:

a) A neural network can perform tasks that a linear program cannot.
b) When an element of the neural network fails, it can continue without any problem by their parallel nature.
c) A neural network learns and does not need to be reprogrammed.
d) It can be implemented in any application.

e) It can be implemented without any problem.

1.9 Disadvantages:

a) The neural network needs training to operate.
b) The architecture of a neural network is different from the architecture of microprocessors therefore needs to be emulated.
c) Requires high processing time for large neural networks.

2. Optical Interconnection Networks

Advances in electro-optic technologies have made optical communication a good networking choice for the increasing demands of high channel bandwidth and low communication latency of high-performance computing/communication applications. Fiber optic communications offer a combination of high bandwidth, low error probability, and gigabit transmission capacity. Multistage Interconnection Networks (MINs) are very popular in switching and communication applications and have been used in telecommunication and parallel computing systems. But these days with growing demand for bandwidth, optical technology is used to implement interconnection networks and switches. In electronic MINs electricity is used, where as in OMINs light is used to transmit the messages. The electronic MINs and the optical MINs have many similarities, but there are some fundamental differences between them such as the optical-loss during switching and the crosstalk problem in the optical switches.

MINs have been an attractive interconnecting structure for high performance parallel computing systems. Available OMINs were built mainly on banyan or its equivalent e.g. baseline, omega networks because they are fast in switch setting self-routing and also have a small number of switches between an input-output pair. Banyan networks have a unique path between an input-output pair, and this makes them blocking networks. Non blocking networks can be constructed by either appending some extra stages to the back of a regular banyan network. To transfer messages from a source address to a destination

address on an optical network without crosstalk, we need to divide the messages into several groups, and then deliver the messages using one time slot for each group, which is called the time division multiplexing (TDM) In each group, the paths of the messages going through the network should be crosstalk free.

Crosstalk in optical networks is one of the major shortcomings in optical switching networks[11], and avoiding crosstalk is an important for making optical communication properly. To avoid a crosstalk, many approaches have been used such as time domain and space domain approaches. Because the messages should be partitioned into several groups to send to the network, some methods are used to find conflicts between the messages.

2.1 Multistage Interconnection Networks

MINs consist of more than one stages of small interconnection elements called switching elements and links interconnecting them. MINs are used in multiprocessing systems to provide cost-effective, high-bandwidth communication between processors and/or memory modules. A MIN normally connects N inputs to N outputs and is referred as an NxN MIN. The parameter N is called the size of the network. There are several different multistage interconnection networks proposed and studied in the literature. Figure 8 illustrates a structure of MIN, which are representatives of a general class of networks. The figure 8 shows the connection between p inputs and b outputs, and connection between these is via number of stages. Multistage interconnection network is actually a compromise between crossbar and shared bus networks of various types of multiprocessor interconnections networks. Multistage interconnection networks are:

- Attempt to reduce cost.
- Attempt to decrease the path length.

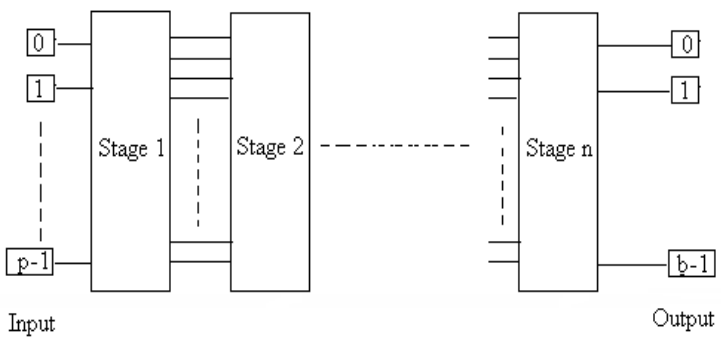


Figure 8. A Multistage Network

Table 1. Properties of different Networks

Property	**Bus**	**Crossbar**	**Multistage**
Speed	Low	High	High
Cost	Low	High	Moderate
Reliability	Low	High	High
Configurability	High	Low	Moderate
Complexity	Low	High	Moderate

2.2 Optical Multistage Interconnection Networks

An OMIN can be implemented with either free-space optics or guided wave technology. It uses the Time Division Multiplexing. To exploit the huge optical bandwidth of fiber, the Wavelength Division Multiplexing (WDM) technique can also be used. With WDM the optical spectrum is divided into many different logical channels, and each channel corresponds to a unique wavelength. Optical switching involves the switching of optical signals, rather than electronic signals as in conventional electronic systems. Two types of guided wave optical switching systems can be used. The first is a hybrid approach in which optical signals are switched, but the switches are electronically controlled. With

this approach, the use of electronic control signals means that the routing will be carried out electronically. As such the speed of the electronic switch control signals can be much less than the bit rate of the optical signals being switched. So, with this approach there is a big speed mismatch occurs due to the high speed of optical signals. The second approach is all-optical switching. This has removed the problem that occurred with the hybrid approach. But, such systems will not become practical in the future and hence only hybrid optical MINs are considered. In hybrid optical MINs, the electronically controlled optical switches, such as lithium neonate directional couplers, can have switching speeds from hundreds of picoseconds to tens of nanoseconds.

2.3 Switching in Optical Networks

In optical networks, circuit switching is used. Packet switching is not possible with OMINs. If packet switching is used, the address information in each packet must be decoded in order to determine the switch state. In a hybrid MIN, it means it require conversions from optical signals to electronic ones, which could be very costly. For this reason, circuit switching is usually preferred in optical MINs. So we assume that circuit switching is used.

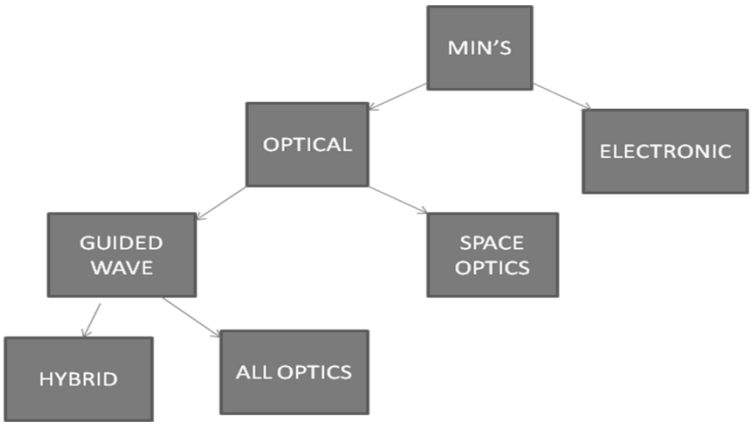


Figure 9. Types of Multistage Networks[1]

There are lots of benefits of optical networks over the electronic ones. The main benefit of the optical networks over the electronic network is the high speed of the Optical signals. In the Optical networks light is transmitted which has a very good speed but in the electronic Multistage interconnection networks electricity is used which has very slow speed. The second advantage is the bandwidth. These days applications in communication require high bandwidth. The optical networks give combination of very high bandwidth and low latency. That is why they have been used in the parallel processing applications. OMINs are also used in wide area networks which require less error probability and very high bandwidth. Fibre optic transmission distance is significantly greater than the electronic ones. The signal needs not to be regenerated in optical networks. Optical fiber has very less weight in comparison to electronic MINs. Thus Optical networks give the combination of high bandwidth and low latency.

Table 2. Difference between electronic and optical networks

2.4

Characteristic	Electronic Multistage Networks	Optical Multistage Networks
Speed	Less	High
Energy Transmitted	Electricity	Light
Bandwidth	Used for less bandwidth applications	Used for high bandwidth applications
Latency	High	Less
Error Probability	High	Less
Weight	More	Less
Cost	Less	More
Switching	Packet Switching	Circuit Switching
Path	Provide Multi path from source to destination.	Provide single path from source to destination
Complexity	More Complex	Less Complex
Structure considered	2-dimensional	3-dimensional

Problems in Optical Networks

Due to the difference in speeds of the electronic and optical switching elements and the nature of optical signals, OMINs also hold their own challenges.

2.4.1 Path Dependent Loss[1,4,5]

Path dependent loss means that optical signals become weak after passing through an optical path. In a large MIN, a big part of the path-dependent loss is directly proportional to the number of couplers that the optical path passes through. Hence, it depends on the architecture used and its network size. Hence, if the optical signal has to pass through more number of stages or switches, the path dependent loss will be more.

2.4.2 Optical Crosstalk[4,5]

Optical crosstalk occurs when two signal channels interact with each other. There are two ways in which optical paths can interact in a switching network. The channels carrying the signals could cross each other. Alternatively, two paths sharing a switch could experience some undesired coupling from one path to another within a switch. For example, assume that the two inputs are y and z, then the two outputs will have ly+lxz and lz+lxy respectively, where l is path loss and x is signal crosstalk in a switch. Using the best device x=35 dB and l=0.25 dB. For more practically available devices, it is more likely that x=20 dB and l=1 dB [16]. Hence, when a signal passes many switches, the input signal will be distorted at the output due to the loss and crosstalk introduced on the path.

Crosstalk problem is more dangerous than the path-dependent loss problem with current optical technology. Thus, switch crosstalk is the most significant factor that reduces the signal-to-noise ratio and limits the size of a network. Luckily, first-order crosstalk can be eliminated by ensuring that a switch is not used by two input signals simultaneously. Once the major source of crosstalk disappears, crosstalk in an optical MIN will have a very small effect on the signal-to-noise ratio and thus a large optical MIN can be built and effectively used in parallel computing systems.

2.5 Ways to Solve Crosstalk Problem[5]

2.5.1 Space Domain Approach

One way to solve the crosstalk problem is a space domain approach, where a MIN is duplicated and combined to avoid crosstalk. The number of switches required for the same connectivity in a network with space domain approach is slightly larger than twice that for the regular network. This approach uses more than double the original network hardware to achieve the same. Thus for the same permutation the hardware or we can say the number of switches will be double. Thus cost will be more with the networks using space domain approach. In the entire four cases only one input and only one output is active at a given time so that no cross talk occurs. With the space domain approach, extra switching elements (SEs) and links are used to ensure that at most one input and one output of every SE will be used at any given time.

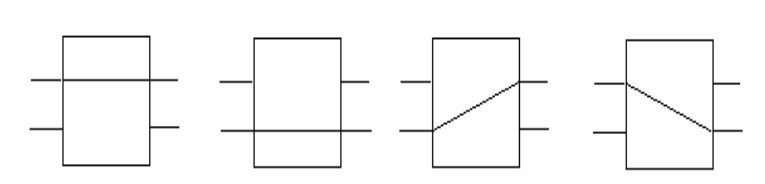


Figure 10. Ways to Avoid Crosstalk in the Network using Space Domain Approach.

2.5.2 Time Domain Approach

Another way to solve the problem of crosstalk is the time domain approach. With the time domain approach, the same objective is achieved by treating crosstalk as a conflict, that is, two connections will be established at different times if they use the same SE. Whereas to distribute the messages to be sent to the network into several groups, a method is used to find out which messages should not be in the same group because they will cause crosstalk in the network. A set of connections is partitioned into several subsets such that the connections in each subset can be established simultaneously in a network. There is no crosstalk in these subsections. This approach makes importance in OMINs for various reasons [1,6,17].

- First, most of the multiprocessors use electronic processors and optical MINs. There is a big mismatch between the slow processing speed in processors and the high communication speed in networks carrying optical signals.

- Second, there is a mismatch between the routing control and the fast signal transmission speed. To avoid crosstalk, we use the TDM approach, which is to partition the set of messages into several groups such that the messages in each group can be sent simultaneously through the network without any crosstalk.

Chapter 2 LITERATURE SURVEY

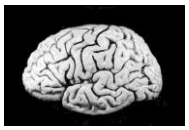
1. The Brain as an Information Processing System

The human brain contains about 10 billion nerve cells, or neurons. On average, each neuron is connected to other neurons through about 10,000 synapses. The actual figures vary greatly, depending on the local neuroanatomy. The brain's network of neurons forms a massively parallel information processing system. This contrasts with conventional computers, in which a single processor executes a single series of instructions.

Against this, consider the time taken for each elementary operation: neurons typically operate at a maximum rate of about 100 Hz, while a conventional CPU carries out several hundred million machine level operations per second. Despite of being built with very slow hardware, the brain has quite remarkable capabilities:

- Its performance tends to degrade gracefully under partial damage. In contrast, most programs and engineered systems are brittle if you remove some arbitrary parts, very likely the whole will cease to function.
- It can learn from experience.
- This means that partial recovery from damage is possible if healthy units can learn to take over the functions previously carried out by the damaged areas.
- It performs massively parallel computations extremely efficiently. For example, complex visual perception occurs within less than 100 ms, that is, 10 processing steps.
- It supports our intelligence and self-awareness. (Nobody knows yet how this occurs.)

Table 3. Comparison between brain and digital computer

	processing elements	**element size**	**energy use**	**processing speed**	**style of computation**	**fault tolerant**	**learns**	**intelligent, conscious**
	10^{14} synapses	10^{-6} m	30 W	100 Hz	parallel, distributed	Yes	yes	Usually
	10^{8} transistors	10^{-6} m	30 W (CPU)	10^{9} Hz	serial, centralized	No	a little	not (yet)

As a discipline of Artificial Intelligence, Neural Networks attempt to bring computers a little closer to the brain's capabilities by imitating certain aspects of information processing in the brain, in a highly simplified way.

1.1 Neural Networks in the Brain

The brain is not homogeneous. At the largest anatomical scale, distinguish cortex, midbrain, brainstem, and cerebellum. Each of these can be hierarchically subdivided into many regions, and areas within each region, either according to the anatomical structure of the neural networks within it, or according to the function performed by them.

The overall pattern of projections (bundles of neural connections) between areas is extremely complex, and only partially known. The best mapped and largest system in the human brain is the visual system, where the first 10 or 11 processing stages have been identified. We distinguish feedforward projections that go from earlier processing stages (near the sensory input) to later ones (near the motor output), from feedback connections that go in the opposite direction.

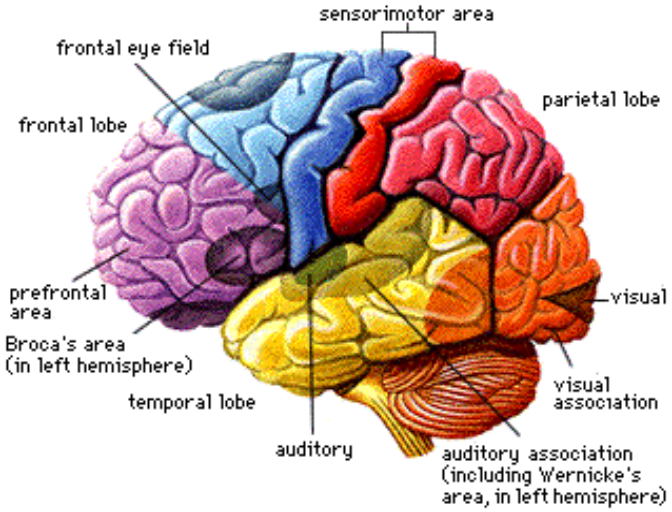


Figure 11. Brain[60]

In addition to these long-range connections, neurons also link up with many thousands of their neighbours. In this way they form very dense, complex local networks.

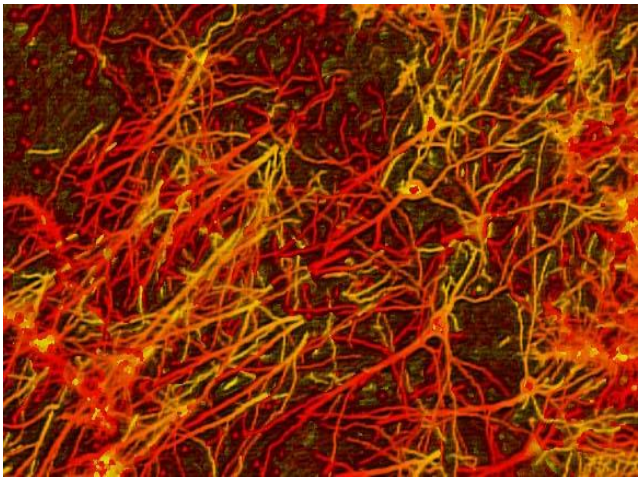


Figure 12. Neurons[60]

1.2 Neurons and Synapses

The basic computational unit in the nervous system is the nerve cell or neuron. A neuron has:

- Dendrites (inputs)

- Cell body
- Axon (output)

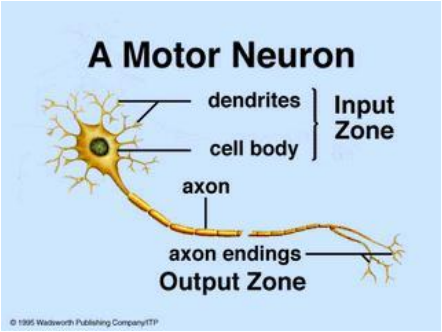


Figure 13. A Motor Neuron

A neuron receives input from other neurons (typically many thousands). Once input exceeds a critical level, the neuron discharges a spike - an electrical pulse that travels from the body, down the axon, to the next neurons or other receptors. This spiking event is also called depolarization, and is followed by a refractory period, during which the neuron is unable to fire.

The axon endings (Output Zone) almost touch the dendrites or cell body of the next neuron. Transmission of an electrical signal from one neuron to the next is effected by neurotransmittors, chemicals which are released from the first neuron and which bind to receptors in the second. This link is called a synapse. The extent to which the signal from one neuron is passed on to the next depends on many factors, e.g. the amount of neurotransmittor available, the number and arrangement of receptors, amount of neurotransmittor reabsorbed etc.

1.3 Synaptic Learning

Brains learn. From what we know of neuronal structures, one way brains learn is by altering the strengths of connections between neurons, and by adding or deleting connections between neurons. Furthermore, they learn on-line, based on experience, and typically without the benefit of a benevolent teacher.

The efficacy of a synapse can change as a result of experience, providing both memory and learning through long-term potentiation. One way this happens is through release of more neurotransmitter. Many other changes may also be involved.

Long-term Potentiation:

> An enduring (>1 hour) increase in synaptic efficacy that results from high-frequency stimulation of an afferent (input) pathway.

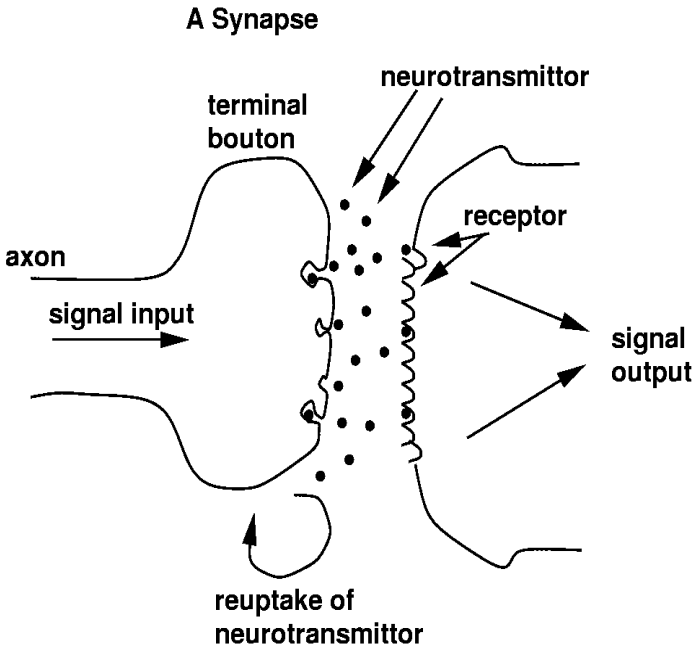


Figure 14. Synapse[60]

2. Software Engineering methods for Neural Networks

2.1 The Problem of Creating Neural Networks for Specific Applications

While there have been numerous successful neural network applications, the development processes reported in the majority of the literature could not be characterised as predictable nor repeatable, both properties central to an engineering

approach. It is the lack of a design process and the absence of verification and validation methods for the design prior to implementation which constitute the most significant contribution to the unpredictable and unrepeatable nature of neural network development.

2.1.1 The Design Problem

In traditional software engineering development, design is a human guided process of developing a solution by choosing, creating and assembling elements to satisfy the users requirements. The elements are sometimes generic such as design patterns and sometimes specific to the given requirements. A program is then a collection of data structures and algorithms chosen during design. Making appropriate choices during design is made easier by layering choices at different levels of abstraction, for example by employing architectural and intermediate design levels. Confidence in the design can be attained through inspections, walkthroughs, prototyping and other similar forms of design evaluation. This approach to design is not applicable for neural network software development because there are no known methods to completely design a neural network a priori. Instead neural networks modify their own structure through the application of learning algorithms. What most often occurs in practice is that a generic neural network model such as a multi layer feed-forward network is combined with a learning algorithm such as backpropagation and problem data to create a specific neural network. The choice of a learning algorithm effectively replaces the more traditional design step, but the process of applying the learning algorithm is not completely automatic and does require human intervention. Further, the selection of parameters for the learning algorithm is closer to design in the traditional sense. For neural networks, however, the verification and validation methods, and guidelines for the selection of parameters are not well developed. There is no oracle or theory which explains how to make the myriad of choices required to apply a generic neural network model to a specific problem.

The wide range of choices available to neural network developers has already been identified as an issue in [12]. The problem of training a machine learning model from data and investigate the issues involved in solving real-world prediction problems. They

use the term application development to describe the overall process of applying a particular model to a domain-specific real-world problem. We extend the concept of application development to problem specification and neural network creation.

Problem specification is the process of selecting the parameters associated with the neural network structure and the learning algorithm. Multiple combinations of selections can be made each combination is referred to as parameter collection. Neural network creation is the process of systematically applying the learning algorithm with each associated parameter collection as defined in the problem specification.

The problem specification and neural network creation steps help to achieve process predictability and repeatability. The resources required for a single iteration of the neural network creation step using a single parameter collection can be measured. In turn the data from these measurements provide us with a basis for predicting the overall resources required to apply the learning algorithm with all of the parameter collections. If historical data from previous projects is available, it become helpful in making prediction before neural network creation begins. By recording all of the parameters associated with neural network creation, its easy to repeat the development by following and applying the learning algorithm with the parameter collections recorded in the problem specification document[12].

The problem specification document is the key mechanism for identifying all of the choices facing developers and structuring these choices according to various levels of abstraction. It contains information on relevant problem domain factors: for example, noise in the collected data, data factors, for example data normalisation and problem encoding; and human factors, for example performance requirements. The problem specification also contains information on the selection of particular neural network structures, learning algorithms, and associated parameters.

2.1.2 The Quality Assurance Problem

The major issue of quality assurance surrounding neural network software is the verification and validation of the generic model - how can be confident about the generic components (e.g. neural network models and learning algorithms) that they are correct.

This raises an issue concerning the appropriate separation of the specific problem requirements and the generic components for a specific problem. The models of neural networks are common across problems but the specifics of the problem are unique to each particular development. The approach to addressing this issue is in providing methods for adequately capturing both the specific and generic requirements.

However, in contrast to the problem specification which is unique to each development effort, the generic neural network model can be reused. Indeed, there are a number of advantages in the neural network development community sharing a generic neural network model.

a) Firstly, a common model allows for problems with the model to be resolved by the community rather than by an individual.
b) Secondly, the validation of the model can be conducted via a process of wide spread community review resulting in a standardized neural network model.
c) Finally, a shared model implemented soundly in a prototype could be used to verify the neural network implementation of any neural network developer.

For example, achieving the same output from the prototype and the given implementation increases the confidence that the given implementation is correct.

Others have attempted to specify neural networks in [35,36,51] but without the benefit of a comprehensive development process, and without addressing the matter of systematic analysis of the problem data or verification and validation issues. Some neural network specific verification and validation work has been addressed in but is limited to benchmarking and does not treat generic and specification concerns. Related work on approaches to using formal specifications to verify and validate knowledge-based systems are presented in [48].

2.1.3 Engineering Neural Networks

The application of software engineering principles to the development of neural networks has received little attention, except in the work of [16] and indirectly in [12]. Its aim is to provide some of the methods required to address the problems in neural network development presented above, and to allow the development of neural networks for specific problems to be incorporated into traditional, planned, software engineering projects. In the literature the process of developing neural networks has typically started with requirements and then moved to implementation, often leaving design choices implicit. However, making these design choices explicit has a number of advantages. In section (2.2), an approach has been discussed which aims to enumerate the set of design choices and can be used to observe correlations between particular choices and resulting development outcomes. By recording design choices, repeatability is possible because all of the information required to repeat the development is documented.

2.2 Factors in the Design of a Neural Network

The design of neural networks is a process of making choices possibly creative choices, regarding the models, learning algorithms and associated parameters. The first set of choices that we typically face in designing a neural network to solve a specific problem is to decide on:

a) The Neural Network Model, for example, multiple layer feed forward networks, feedback networks, radial basis function networks, associative memories, support vector machines and the type of transfer function the nodes in the network implement.

b) The Learning Algorithm, which often depends on the choice of neural network model, but can be divided into:

i) Gradient descent methods, such as back propagation and associated derivative algorithms like quickprop and robustprop.

ii) Stochastic methods, such as simulated annealing, genetic algorithms, evolutionary programming and monte carlo methods.

A complicating factor is that unless there is specific guidance available then typically need to choose a number of different combinations of neural network models and

learning algorithms in order to determine which combination will satisfy the performance requirements. By documenting these choices and the results for specific problems we can begin to collect enough empirical knowledge to provide guidance in making these choices.

The next step is to choose the parameters associated with our first set of choices. Current models and learning algorithms utilise the following parameters:

a) parameters governing the structure of the neural network include:

- the number of layers
- the number of nodes in each layer
- the transfer function in each node
- the weights and their initial settings which determine the connectivity between nodes

b) learning algorithm parameters — each learning algorithm has a number of parameters associated with it, such as the learning constant and the momentum factor for back propagation.

c) training and testing strategy—what proportion of patterns should be used for training and what proportion for testing?

d) training and testing patterns — which patterns, out of all of those available, should be used for testing and training?

e) error levels — what error levels, both in terms of precision and accuracy, can be expected?

The selection of appropriate parameters is the difference between solving the given problem and not solving it. A major dilemma facing the neural network software developer is what set of parameters should be used for a particular problem? At this stage, no oracle or body of empirical data is available to answer this question. Several attempts to automate this process have been proposed, particularly those which employ evolutionary computing methods [38,42].

2.3 The Process of Developing Neural Networks

2.3.1 Predicting and Repeating Development

There is an excellent reason for recording relevant problem information. By defining the datasets and testing and training strategy in the problem specification phase, the neural network creation phase can be prevented from consuming an unbounded and unplanned amount of resources. Another major problem facing neural network development is the unpredictable nature of current practice with regard to the creation of a specific neural network. The current approach can be characterised as an unguided iterative approach along the lines of:

a) Choose a problem representation, create an associated data set and set performance requirements.
b) Train a neural network while varying the associated parameters to achieve the given performance requirements.
c) If the performance requirements are met, stop. Otherwise try again.

The problem with this approach is that there is no plan for meeting the requirements and the number of iterations is often unbounded. It is particularly hard to isolate the reason for not achieving the performance requirements, was it the parameters associated with the testing and training strategy? Or the problem representation?

Our solution is to break the activities into distinct phases which cannot overlap. This does not stop feedback between the phases but prevents the unbounded consumption of resources in the neural network creation phase. An example of this process is:

a) Create N valid according to the developers judgement problem representations and create N associated data sets.
b) Associate each data set with a testing and training strategy and all the parameters required for neural network development.
c) Rank each collection of data sets, strategy and associated parameters for the neural network creation phase. This ranking places an order on which collection will be used to develop a specific neural network first.
d) Beginning with the highest ranking collection, develop a neural network in accordance with the data set, strategy, and parameters supplied.

e) If the performance requirements are met, stop, otherwise proceed to the next collection. If all collections are tried and still none of the performance constraints are met then the neural network development should be reevaluated.

This rest of this section is devoted to a more detailed presentation of the process model for neural network development.

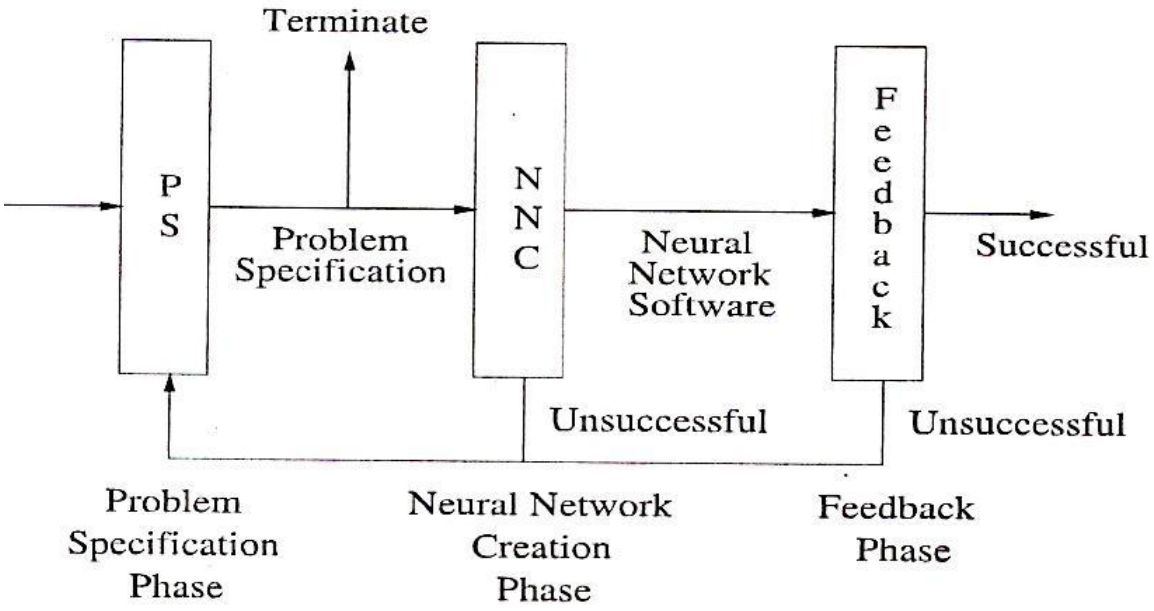


Figure 15. Neural Network Development[30]

At the highest conceptual level there are three distinct phases as shown in figure 15.

a) Problem Specification
b) Neural Network Creation
c) Feedback

However, before providing details of the development process must be need to identify the roles associated with development.

In a generic description of software development, there are at least three roles: a user, a client and a developer. These correspond to roles in neural network development.

a) Firstly, the neural network user, is the person who uses neural network software to solve problems as part of an overall system.

b) Secondly, the nature of neural networks requires an understanding of the problem that is being solved. The role with this understanding is referred to as the problem domain expert and roughly corresponds to a client.

c) Thirdly, there is the neural network developer. This is an individual or group who creates specific neural network software for a neural network user using generic neural network software.

In addition to these three roles, there is an additional role involved in neural network software development, the generic neural network developer. It is their role to create the generic neural network software based on neural network models. The roles and their relationships are illustrated in figure 16.

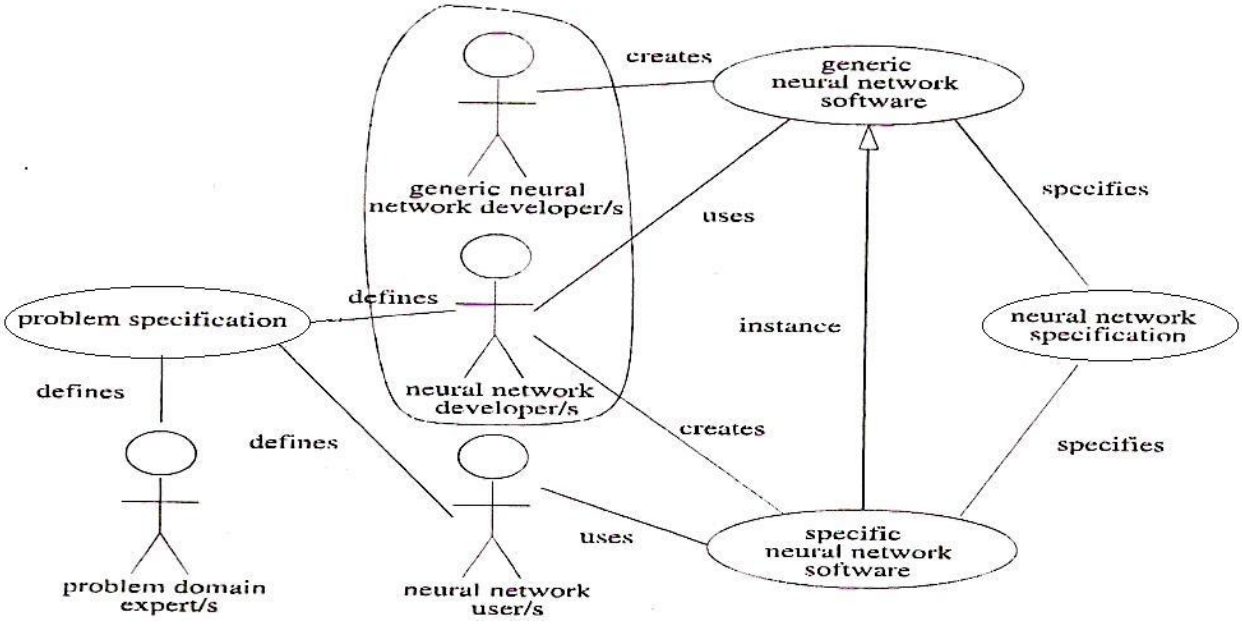


Figure 16. Development Roles[30]

2.3.2 The Problem Specification Phase

The first phase is problem specification where the problem (as determined by the associated data) is thoroughly defined, as are analysis techniques and precision and accuracy requirements. Problem specification is performed by the user, developer and problem domain expert/client. A majority of this information is used in the next phase of

development but some is used to index and characterise the problem for later use as part of a historical record for future development and planning, to allow for repeatability.

There are three main scenarios concerning the specification of neural network development requirements.

a) Firstly, where the client requires that, for some aspect of the system, a neural network is used. In this case, the neural network is a component of a larger system and the accuracy and precision constraints of the neural network are explicitly given by the client.
b) Secondly, where the client requires some functionality which has associated performance requirements which can be satisfied by using a neural network. This is similar to the first case except that the client has no explicit requirement for using a neural network but the developer determines that a neural network could be an appropriate technology to satisfy the clients requirements and users needs.
c) Thirdly, where the client requires some functionality which could be satisfied by using a neural network but does not have any performance requirements. In this case, the developer must set appropriate performance constraints and can identify alternative technologies (for example, Bayesian belief networks or genetic algorithms) for potential use if neural network development is unsuccessful. The performance requirements are implicit in the design of the system determined by the developer. The dual nature of neural network requirements means a specification of both the problem data and neural network models is needed. This section addressing the problem specific content. The key property of the mechanism for recording the problem specific content is repeatability. However, because of the lack of theory associated with neural network creation, the exact information to record depends on the specific problem. To accommodate this variation, a problem specification document template which can be adapted on a case by case basis is used.

The problem specification document captures the problem specific requirements in sufficient detail to allow for repeatability of development. The problem specification document is not complete once the problem specification phase has finished. The results

of neural network creation are included in the document for predictability and repeatability purposes. Even unsuccessful neural network training is documented to ensure that unsuccessful development is avoided the next time a similar problem is faced. The key content in the problem specification document are the collections which contain the different problem representations, learning parameters, precision and accuracy requirements and testing and training strategies. These collections give explicit guidelines for creating the specific neural network. Each of the collections is ranked by the likelihood that the collection will achieve the precision and accuracy requirements. The method for predicting the likelihood of achieving given precision and accuracy requirements is based on individual experience. As more experience is gathered within the neural network development community a more suitable method can be developed.

2.3.3 The Neural Network Creation Phase

The second phase is that of neural network creation, where the neural network software is created by combining the collections specified in the problem specification document and the generic neural network technology. As part of this phase, a specification of the generic neural network technology is needed. Such a specification addresses the second part of the previously identified requirements problem. Three desirable properties of a neural network specification are formal verification. validation methods, and easy implementation and prototype creation.

2.3.4 The Feedback Phase

The third and final phase, feedback, is where the client and user evaluate the performance of the neural network. Depending on the capabilities of the development organisation, alternative methods can be implemented for evaluation purposes. Alternative methods provide a sanity check for the neural network solution that has been developed. These activities establish whether the performance of the neural network is better than available alternatives. The resulting problem specification, project plan, neural network formal

model used for development, and the feedback from the user and evaluation of the neural network are placed in the historical record for repeatability and planning purposes.

2.4 Phases of Development

2.4.1 Problem Specification

The aim of the problem specification phase is to specify the problem, analyse the quality of the problem data and specify performance requirements. These activities are supported by a problem specification document template. The template includes sections for data analysis, data representation, and methods for data preparation as well as specifying the performance requirements. The performance requirements stipulate the desired precision and accuracy of the specific neural network software for it to be considered successful. Without these requirements, determining when the neural network creation phase has concluded is problematic and can lead to excessive expenditure of development resources.

The main reason for performing data analysis early in the development process is to provide an assessment of the viability of solving the problem before neural network creation begins[30]. Neural network learning is a data driven technique and for this reason the data used to create the neural network is of critical importance. Unfortunately, neural networks learning algorithms are not so sophisticated as to identify poor data in themselves. Neural networks are robust in the face of poor quality data but it is advisable to assess the data before neural network creation. Creating a neural network with higher quality data increases the likelihood of learning the problem. The difficulty is in deciding if data is actually poor quality or if it only appears to be but this is indicative of the problem-domain. There is a trade-off here, making the data easier to learn can be at the expense of eliminating valuable, possibly even critical, data. This trade-off must be made on a case by case basis but it is clear that recording these decisions for repeatability purposes and future tradeoffs is a good idea. Refer the reader to the following sources on specific data analysis techniques:

a) Standard data analysis methods, such as, re-scaled range analysis, randomness tests, sampling error estimates, principal component analysis.
b) Standard visualisation measures, for example, the phase and frequency of the data.
c) Statistical summaries, such as the mean, median, mode, mean deviation and standard deviation. These measures of data sets capture key features of central tendency and spread and have nice properties under suitable conditions [9].
d) Graphical summaries in the form of histograms, stemand-leaf plots and box-plots provide useful information about the centre, spread and other features of a dataset [9].

In the case of problems for which there are few or no similar examples the historical record may be inadequate. The suggestion is to replace the historical record with research into the problem domain.

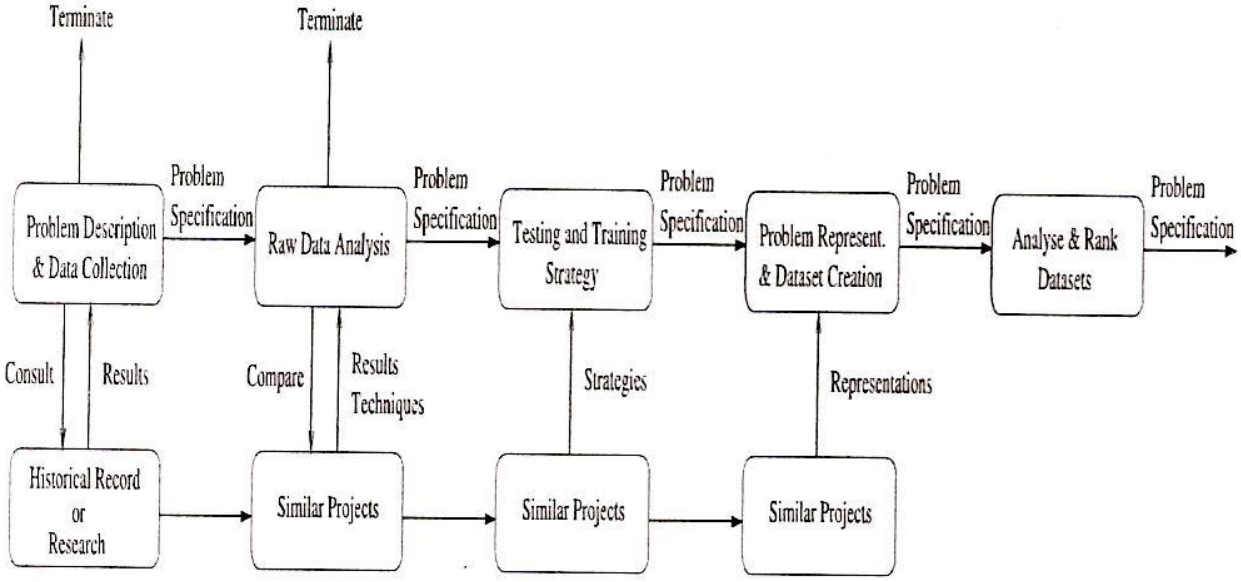


Figure 17. Problem specification phase[30,43]

Note that some issues are hidden within this process. For example, in one application in mathematical formula recognition, no recognition error levels were supplied by the user and error levels were determined through data analysis .

2.4.2 Neural Network Specification

The hardest single part of building a system is deciding what to build. Sallis et. al. estimate that approximately 53% of all problems in software development result from poor requirements [26]. Schach et. al. report results from a variety of projects, conducted over the last 25 years, which describe the rapid increase in cost for correcting faults originating from poor requirements as software development progresses. Indeed, there is universal agreement that poor requirements are a significant cause of problems later in the software development process. Given the pivotal role of the generic neural network software we require that the associated requirements are less problematic. One form that requirements can be stated in is a formal model.

The advantages of formal models for requirements specification have been well documented [8,22,53]. Firstly, they provide an environment suitable to analyse requirements and design in order to explore the consequences of choices and resolve problems. Typically, a formal model can be used to find ambiguous or incomplete requirements, determine over-specification or predict some aspects of behaviour. This can be accomplished by proving properties of the model or creating a prototype from the model and testing its properties. In particular, rapidly creating a prototype is an inexpensive method for gaining confidence that a specification model is complete, consistent and captures the key properties required of the proposed system (an example of domain directed verification). Prototyping and its relation to the verification and validation of the neural network requirements is the subject of a later section.

Recall in figure 16 that there are two key parties concerned with the neural network requirements, at different levels of abstraction, are the neural network user and the neural network developer. The user is typically concerned with the requirements at a higher level of abstraction (such as the dynamic operation) while the developer is concerned with requirements at all levels of abstraction. Both sets of requirements are presented separately but we show how the two sets of requirements can be kept consistent within a single model[16,23,24,28,34].

The user requirements for a neural network as a black box which takes a certain number of input and produces a certain number of output[8,21,29].

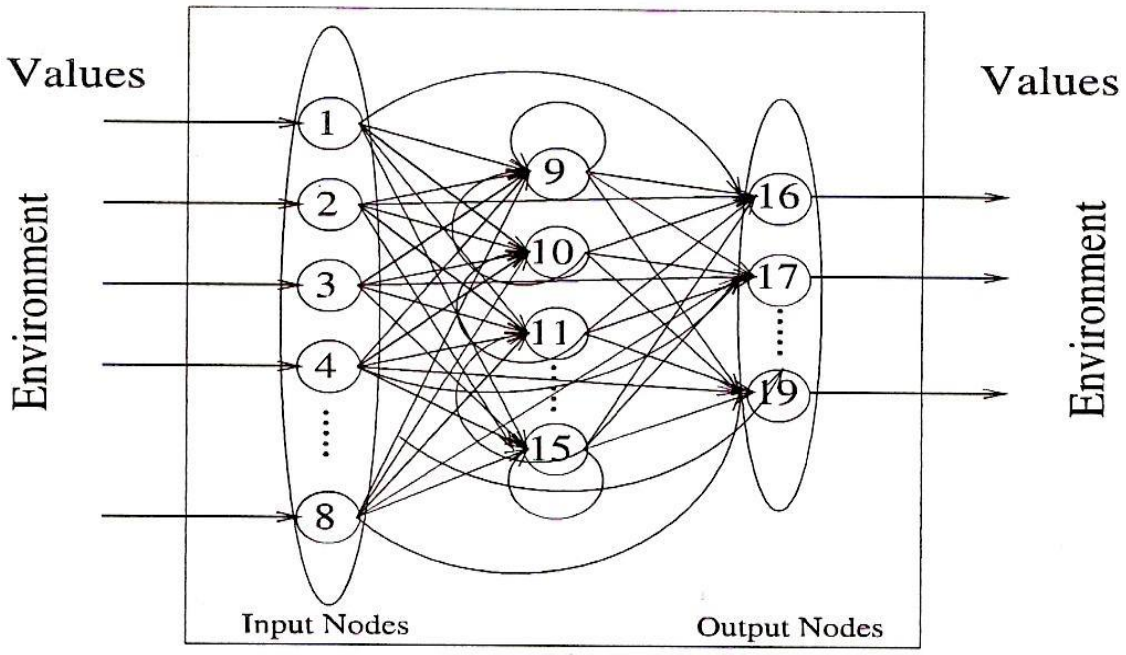


Figure 18. Model of basic network[30]

In order to demonstrate the usefulness of the method of specification now specify a basic neural network and use this as the basis for defining a layered network through the addition of constraints. By this method, the user and developer requirements are consistent even though they are described at different levels of abstraction. Common properties and structure of a basic neural network is:

a) Every network has a sequence of input nodes which can receive stimulus (in the form of values) from the environment, ie. values from outside the network. In the schema we denote input nodes by input nodes.
b) Every network has a sequence of output nodes which transmit values from the network to the environment. In the schema we denote output nodes by output nodes.

c) Each node in a network has an identifier called a nodeid. This allows the specification to define neural networks without knowledge of the node. It also means that networks can be created with a variety of nodes so long as they have a common interface. In this way a heterogeneous network of nodes have been created. That is, a network must be able to have a mixture of nodes with different transfer functions.
d) Every network contains a function, connections, which defines the connectivity of nodes in the network. The connections store the weights which connect nodes together.
e) In all networks, each node has a sequence ordered set of source nodes which supply excitory or inhibitory values. In order to be able to apply an inverse mapping of the sources, a sinks function an injective sequence have been used.

Also identify several building block components: nodes, values, nodeids, and weights. Nodes are a special type of component which is a parameter of a given network. The basic neural network specification factors out all the common properties neural networks share. A definition of the sets of possible nodeids, weights and values that a basic network can use is needed before specifying basic network requirements for the developer. This is done using the given sets

[*NODE, VALUE, NODEID, WEIGHT*]

With these definitions made a basic network can be specified. A major advantage of our specification approach is the ability to add constraints to previously defined schemas to create new schemas.

2.5 Verification & Validation Methods

The production of prototype implementations is potentially a very convincing way to explore the behaviour of a specification with its eventual user, and it is in that combination that suggest formal specification and prototyping offer the most synergy [8]. The approach to verifying neural network models is through constructing prototype implementations. Prototyping of the formal model allows for the specified properties and

the resulting behaviour to be observed. Confidence in the implementation of the prototype and the model is gained by running appropriate test cases and comparing the results with what is expected. There are two issues that must be addressed when validating formal models using a prototype.

a) Firstly, the prototype must be a faithful representation of the model to allow the relevant parties to test the requirements through the prototype.
b) Secondly, the final delivered system must be a correct implementation of the model and behave in the same way as the prototype.

Without this it is possible to fail to meet the expectations of the user for the system. The use of prototypes to gain confidence in the correctness of formal models has been previously proposed for software and hardware development. Meseguer and Preece advocate the use of formal specifications to verify and validate knowledge-based systems [48]. The relevance here is that neural networks and knowledge-based systems are similar techniques, both used to solve complex, poorly understood problems which are derived from data. Meseguer and Preece use successively more detailed models, starting with a conceptual model then a formal specification and finally an implemented knowledge-based system. The formal specification is internally verified and validated against the conceptual model. Decomposing the verification and validation tasks makes them easier to perform and provides a path from conceptual model to implementation.

The model is verified by generating a prototype and exploring its behaviour. The prototype can be applied to solve known problems as test cases and the responses observed. One of the test cases used to verify the prototype was the XOR problem. For example, to verify the recall operation, the results of recall for a defined XOR network were compared with the results published by Gibb and Hamey. The results obtained from the prototype were identical and instilled a higher degree of confidence that both the model and prototype were correct. Solving more problems will increase the confidence in both the prototype and the formal model and has been carried out.

The most significant advantage of having a prototype derived from the formal model is that it provides an additional executable model against which developers can benchmark their own implementations. The separation of the generic developer requirements and the

specific user requirements is reflected in the prototype[30]. The developer can down-load the prototype and instantiate it with any specific network, execute the resulting prototype and compare the results with those of their own implementation. Another advantage is that the process of translating the formal model into a prototype is valuable for gaining insight into design and implementation issues.

3. A Software Development Process Model for Artificial Neural Networks in Critical Applications

ANNs, until recently, were considered experimental models whose place in mainstream computing was uncertain. With the emergence of new network models, user-fiiendly ANN software, and powerful, inexpensive computers, neural networks are appearing in a variety of applications. Neural networks are fundamentally different from traditional computer models. Whereas algorithmic software is developed through a quantifiable set of steps wherein the program is told exactly what to do, ANNs are constructed by presenting a training program with the information that the network is to know. Since there is no direct control of the training process by the neural net developer, questions arise regarding how to verify that the ANN has learned the correct information. In critical applications, where a malfunction can have catastrophic results, it is particularly important to ensure that the neural network operates correctly.

Given the recent entry of neural nets into mainstream computing, there is a need for a software development process model to guide ANN developers to a more formalized methodology. For neural networks in critical applications, there is a great need for these models to reflect current thinking on risk management. The neural network development process model discussed in this section attempts to satisfy the needs of neural network developers in terms of critical application certification, risk minimization, and usability.

3.1 Critical Application Issues and Formal Methodologies

Critical applications fall into a number of different categories, depending on the context of the problem domain. The military often refers to mission critical systems, whose failure will potentially cause an inability to complete some mission - local or global, minor or major. Industrial mission critical systems are related to an organization's ability to do business. Safety critical applications might include traffic signal controls, where failure could endanger life or limb. Health critical applications include systems that control treatment or monitoring, and whose failure could directly cause misdiagnosis or improper administration of medicine, radiation, or therapy leading to immediate or eventual injury or death. Many other critical applications certainly exist in other fields. A common thread that runs through critical applications is that their failure causes problems that go beyond inconvenience. Thus there is an additional motivation to ensure the proper design and construction of critical systems beyond more mundane applications.

3.2 Neural Network Development

An argument could be made that any system, critical or not, should be subjected to a rigorous development standard. There are two compelling reasons why this is uncommon with ANNs.

a) First, due to the imprecise nature of neural network construction and training, development has a distinctly empirical or experimental flavor.
b) Second, the intelligence of neural networks is contained in a collection of numeric synaptic weights, connections, transfer functions, and other network definition parameters.

In general, inspection of these quantities yields little explicit information to enlighten developer as to why a certain result is being produced. The combination of these two factors often results in formal specifications for neural networks being ignored or nonexistent. Neural network developers often take the approach of doing as well as they can with the data, and when additional effort yields asymptotically diminishing improvement, the system is deemed to be good enough. Clearly, when a difficult to understand system is embedded in a critical application, this approach is simply not good

enough. Regardless of the major differences between neural networks and algorithmic programs, there is no fundamental reason why a formal development methodology and validation technique cannot be derived for ANNs. Indeed, for embedding neural networks into host applications, it can be argued that the simple nature of the ANN recall module would make it much easier to certify than the complex code that surrounds it. Neither the common 'Waterfall' [58] nor 'Spiral' [10] software life cycle models apply very well to the development of artificial neural networks. Development of a formal set of performance and interface requirements a priori is usually not possible, since there is no obvious clue in most training data about the level of consistency that is contained therein, and data preprocessing usually requires a great deal of trial-and-error iteration before it is clear what data should be passed from the calling function into the ANN. Rather, the following the six phases that ANN developers usually include in their efforts :

a) Formulation of requirements and goals
b) Selection of training and test data sets
c) Selection of the neural network architecture
d) Training of the network
e) Testing of the network
f) Acceptance and use by the customer

Item (a) in particular contains a very subtle detail. Given the uncertainties that are ahays present in neural net training, the up-front goals should be described separately from the firm requirements. Also, ANN training is usually very iterative, and a loop back from (e) to (b) is very common. The goal here, then, is to form this time-tested (but research and development-oriented) methodology into a formal process that can be controlled, tracked, and analyzed. If this can be accomplished, then neural network development can be brought to the same level of accountability that mainstream software is, opening the door to more readily achievable formal certification of neural network projects.

3.3 Neural Network Risk Management

Some efforts are underway to formalize parts of the neural network development process. The Food and Drug Administration has published some broad guidelines regarding steps that should be taken for ANNs embedded in medical devices. This document also supports varying levels of formalism for embedded medical neural network projects based on Level of Concern. More detailed work is underway in the UK on a formal standard for certifying ANNs in safety critical systems. A draft standard to ensure that the ANN derivation process is replicable and unambiguous has been published for review, and is summarized here.

Designing the Network

- High-level goals must be documented.
- Network architecture must be completely specified.
- The ANN must be able to qualify output fidelity across the input parameter space.

Training the Network

- The data set must be consistent with the high-level goals.
- The developer must document the collecting of data.
- Pre- and post-processing activities must be documented.
- The developer must inspect the data to ensure that no large gaps exist in the input parameter space.
- The developer must detail the accuracy of the network in terms of the available training data.

Interfacing the Network

- The developer must specify the required ANN connection to the system for pre- and post-processing and application timing.

Process Certification Requirements

- The development team manager must understand neural networks.
- Team members must each understand their part in the ANN development process.
- ANN recall routines should be developed using accepted formal methods.
- Task decomposition should reflect the ANN development process.
- QA methods must be used in assembling the ANN training database.

Testing the Network

- The V&V team is to independent of the development team.
- The V&V team shall ensure the ANNs ability to generalize the underlying principles contained in the database.
- The V&V team must validate the operation of the ANN in the host application.

Risk Analysis

- The developer must ensure that the ANN is consistent with standard risk assessment methodologies.
- The developer must establish possible ANN failure modes.
- The developer must investigate input error propagation from the calling routine through the network.
- The developer must investigate the effect of error propagation from the network back to the calling routine.
- Risk analysis should be performed independently by the development and V&V team, and their reports should be reconciled before certification.

The requirements set forth above are focused on risks associated with the technical development and performance of the neural networks. Mainstream risk management also includes other types of risk, including cost, schedule, and maintenance. These issues are not of as much concern with ANN development as with more algorithmic projects. In particular, the magnitude of ANN development projects is generally smaller than code-writing projects, and much of the effort expended is with auxiliary tasks such as database analysis and data preparation/pre-processing. Most ANN development projects range from a few person-weeks to a few personmonths. For projects where the neural net is embedded within a larger application, the host application will normally entail the larger part of the development effort. Maintenance risks are likewise minimal for neural nets, as they are not often modified after their deployment.

3.4 The Nested Loop Model of the Neural Network Development Process

The preceeding section discussed current practices widely used in the neural network development community and a review of literature support for risk management in ANN development. This section presents an original concept, similar to existing models for software development. Figure 19 shows a graphical representation of the new process model. It bears more than a passing resemblance to the waterfall model, plus some aspects of the spiral model. Closer inspection of the process elements reveals the fundamental differences needed for neural net development. The pedigree of this process includes existing process models [9], standard ANN development practices, the draft certification standard, and the author's health critical ANN development experience .

Step 1: Network Requirements, Goals, and Constraints[16]

The first major block on the process diagram represents the specification task. It is here that the firm network requirements, goals such as desired accuracy, and constraints (programming language, memory limits, maximum run-time, etc.) are documented. This task should be performed jointly by the ANN developer, the product manager, and the individual responsible for the specification of the calling function. The result of this activity is the Network Performance Specification. This document could be patterned after a down-sized Software Requirements Specification, with a few modifications for the neural network domain.

Step 2: Data Gathering and Preprocessing[16]

The next step in this process is data gathering and preprocessing. Data gathering includes assembling all the data that will be used in training the neural network. These may be found in databases, spreadsheets, other computer files, or printed matter. All activities associated with data preparation should be recorded. This includes the data sources, the original format of the data, the modifications performed to format the data consistently, and any information on the pedigree or fidelity of the data that is available. The preprocessing performed in this step is only to collect the data into an electronic format that is consistent with the neural network training package being used. This step of the process results in the Data Analysis Document, which provides traceability for the training database. This document consists primarily of a log of all data collection and

preprocessing tasks. All data should be attached to this document, along with any programs or macros that were used to assemble it into its usable format.

Step 3: Training and Testing Loops[16]

The center of the process chart contains the element wherein the neural network is actually trained. The Training and Testing Loops represent an iterative process. The idea of nested loops should be familiar to almost all software developers. The application here has for the innermost loop, Variation of ANN Topologies. This denotes changes of intra-paradigm architectural parameters. For multi-layer perceptrons, for example, this would include variations in the number of hidden layers, neurons per layer, activation function choices, bias neuron connection, and similar parameters.

The middle Variation of ANN Paradigms loop contains more basic network modifications. In this loop, the developer might experiment with multi-layer perceptrons, radial basis functions, or many other ANN paradigms. The outermost loop is Selection and Combination of ANN Input Neurons. Changes made here affect the fundamental inputs that the ANN is to model. Quite often, not all available inputs are needed to specify the output space. Othertimes, the inputs will need to be preprocessed to reduce dynamic range, eliminate noise, or somehow time-average the data. In still other cases, non-linear combinations of input variables are pow and indicators, and the neural network training will not uncover them. This type of variation is performed in this outermost loop. The nesting order of the loops was selected such that minor network changes are contained in the innermost loop, moderate changes are in the middle loop, and fundamental network input changes are addressed in the most slowly changing outermost loop. While this seems logical, for some problems the nesting of the loops could be modified by combining the two inner most loops, swapping the two outermost loops, or similar arrangements. Also, some newer neural network training packages automate much of this process and the user has little control once training starts. In such cases, great care must be taken to ensure that the results of the automatic network configuration are not accepted without close inspection[16].

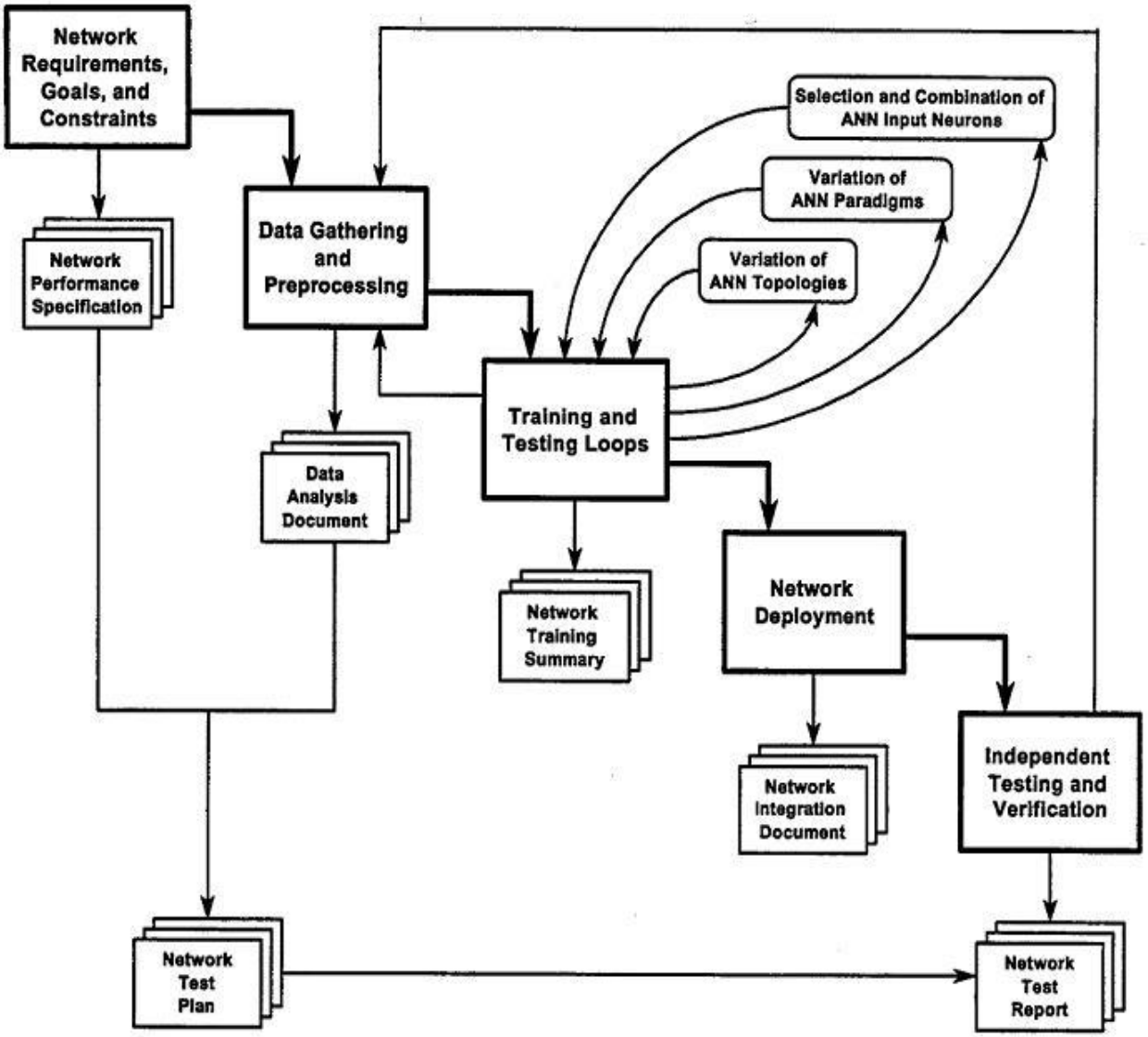


Figure 19. Nested Loop Model of the Neural Network Development Process[16]

During this phase of the process, it should become clear to the developer if the database is complete and selfconsistent. The developer should examine the database during this phase to ensure that the input parameter space is covered sufficiently, that is, to ensure that there are no large gaps in the database. It may also become apparent, if the networks fail to converge, that the problem is underspecified, and additional input parameters need to be investigated. If either of these two situations occur, the developer will need to step back to the Data Gathering and Preprocessing phase for additional work. All developer

activities in training the network should be documented in the Network Training Summary. This document should include a complete list of all permutations that were tried, along with the results achieved. A collection of log-sheets would be sufficient, along with a brief discussion of the general process used in the training phase. All input data files used should be attached electronically, though output and network files from intermediate training sessions are not necessary.

Step 4: Network Deployment[16]

After the developer is satisfied with that the neural net has met its performance requirements and goals, and ANN improvements have reached the point of diminishing returns, the network generally needs to be deployed into a host application. While most commercial neural net training tools include run-time functions, the majority of ANNs are eventually deployed as a module embedded within another program. There are three general ways that this is done.

a) First, many commercial tools include a runtime library that can be linked to the trained network file. In this case, there is little responsibility for the developer other than following the instructions for the linking procedure.
b) The second method is automatic code generation facilities provided by many commercial tools. After the network is trained, source code to compile within the host application can be created in a variety of computer languages, with C/C+ and Visual Basic being among the most popular. In the case of multi-platform languages, care must be taken to customize the code for the particular compiler that is being used for the project.
c) Finally, some commercial programs and most ANN tools that are developed in-house have no facilities beyond saving raw network data to a text file. In this case, the developer must write or otherwise acquire the code to load the ANN data and exercise the network.

Most organizations that do this type of work on a regular basis form a repository of functions for this purpose that can be adapted to any network with minor modifications. The developer should document the deployment effort and record all manually or automatically generated source code in the Network Integration Document. If the

developer needs to create a major deployment module, that part of the code should be placed under the control of the host application's development process. If linked libraries, automatic code generation, or previously accepted in-house libraries are used, the developer should provide references to source documentation of the interfkce. Finally, required data pre- and post-processing requirements for input and output parameters must be explicitly defined, along with an analysis of possible error conditions the ANN could cause.

Step 5: Independent Testing and Verification

When the developer is satisfied that all constraints have been met and all efforts have been documented, the project is handed to the testing group for verification. The first task of the testing group is to construct the Network Test Plan, based largely on the Network Performance Specification and the Data Analysis Document[16]. The test plan should include a critical review of all phases of the development effort, from data gathering forward. The original database should be examined for integrity. The preprocessing should be reviewed for accepted procedures. The training databases should be examined statistically to ensure that the developer did not introduce an unintentional bias in the preprocessing activities. The training and testing phase should be inspected for a rigorous and complete process. Reference [56] present sets of metrics that can be used to assess the accuracy of neural nets. The deployment task should be analyzed to ensure code generation was performed correctly. If any one or more of these phases is found deficient, the testing group has the authority to step the program back to the data gathering phase. Although the process diagram shows a loop only back to data gathering, the project can also be reset to the training and testing or deployment phases, as appropriate. The activities of the testing group, including summaries of all tests performed and electronic copies of any test programs or databases, should be documented in the Network Test Report. This report should include a section on deficiency resolution and retesting, if applicable.

4. Comparing Neural Network with Small World Network

The impressive ability of human and animal brains to recognize and manipulate complex patterns under real-world conditions continues to appeal not only to biologists but also to physics, computer scientists and engineers, albeit with the latter driven by different objectives and motivations[27,32]. Brain activity can indeed be modeled as dynamical process acting on a network, each vertex of the structure represents an elementary component, such as brain areas, groups of neurons or individual cells, each link of the structure represents nerve fiber on which information is transferred from one cell to another.

ANN have been used as a model for simulating brain neural network (BNN), and a considerable amount of work has been made in the field. For example, Hopfield was one of the first to introduce a simple model to describe associative memory in recurrent neural networks successfully, based on the biologically motivated Hebbian rule for adapting the connections between the neurons. The model initiated a period of intense research activity. Most of this work regards both the simulation and the theory of completely connected networks, as well as networks with a random dilution of the connectivity. It is known that particular prescriptions for the determination of the synaptic weights enable these systems to successfully retrieve a pattern out of a set of memorized ones. This behavior is observed in the system up to a certain value of the number of stored patterns, beyond which the network becomes unable to retrieve any of them for reasons of simplicity of the models and their analytical tractability, complex architectures of the networks, more akin to those found in biological neural systems, have been largely left out of the theoretical analysis [32].

The success of these early neural network models was mainly due to their analytic tractability, which was achieved upon sacrificing biological realism, all neurological connectivity structures were sacrificed by the first generation of models [27].

Fortunately, since a few years ago, a class of models that has come to be known as complex networks began to be thoroughly studied [39]. Complex networks seem more compatible with the geometrical properties of many biological and social phenomena than regular lattices, random networks, or completely connected systems.

Many biological neural networks are small-world networks. In most existing literature about small-world networks, there are no weightings in their internal connections of nodes. However, there are many networks, particularly biological neural networks, having weights associated with the connections.

4.1 Comparison between ANN, BNN and SWN[33,34]

In order to understand and improve the structure and function of artificial neural network, firstly make a comparison among artificial neural network (ANN), brain neural network (BNN) and small-world network (SWN).

4.1.1 Comparison between ANN and BNN

The differences of artificial neural network and brain neural network consist in there complex levels: vertices (cells), links (nerve fibers) and function (ability).

a) The first level of complexity is showed by the diversity of vertices (cells). In the ANN, the diversity of vertices means the three different facets of vertices: number, type and threshold. For example, there are three kind vertices in BP neural network: input layer vertices, hidden layer vertices and output layer vertices. Commonly, the hidden layer vertices and output layer vertices have different thresholds and transmit functions. Every vertex has some inputs and one output (figure 20). In the BNN, the diversity of cells means the three different facets of cells: number, type and activity. There are many kinds of cells in brain neural networks, and the components of these cells are different respectively. So it cannot denote these cells using a uniform way. Every cell has a different activity

when something stimulates it. Apparently, the complex degree of cells is much higher than that of artificial neural network.

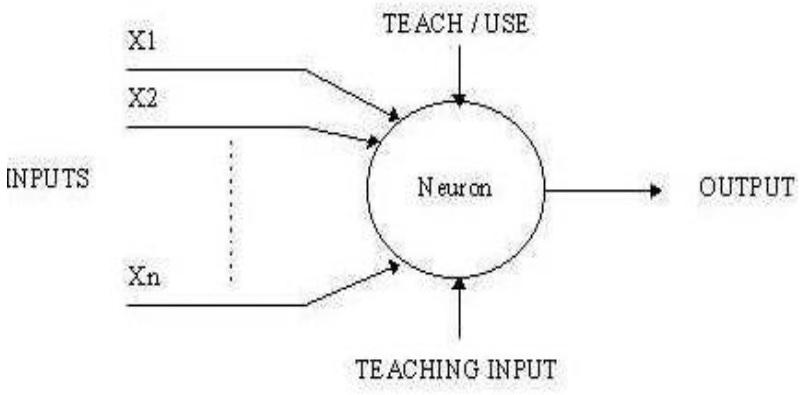


Figure 20. The structure of artificial neural network neuron[15].

b) The second level of complexity is represented by the diversity of links (nerve fibers). The diversity of links in ANN means the three facets of links: number, weight and distribution. There is a small quantity of links generally in ANN. Each link has a weight which is different from another link. At the same time, the distribution of links in ANN is regular. For example, in BP neural network, it is fully connected between layers, and there is a weight on each link. Those weights are different between 0 and 1 respectively. The diversity of links in BNN means the three facets of nerve fibers: number, length and distribution. In brain neural network, it is not fully connected between cells, and the distribution of nerve fibers is not regular but uneven. The length of nerve fibers is varied form short to long. But there is a same point between ANN and BNN. Some kinds of knowledge are also stored in nerve fibers as well as weight on link of ANN.
c) The third level of complexity is showed by the diversity of their function. The ANN can complete memory, association, recognizing, etc. The ability of brain neural network can do more than that of ANN.

Table 4. The comparison between ANN and brain neural network

Complex level	**Artificial neural network (ANN)**	**Brain neural network (BNN)**
Vertices/Cells	Diversity of number, type and threshold	Diversity of number, type and activity
Links/nerve fibres	Diversity of number, weight and distribution	Diversity of number, length and distribution
Function/ability	Diversity of function	Diversity of ability

In summary, the ANN can approximately simulate the structure and function of brain neural network, but the ANN has many defect and limit. So many researcher work hard to improve the structure and function of ANN.

4.1.2 Comparison between BNN and SWN

A great deal of research interest in the theory and applications of small-world networks have arisen since the pioneering work of Watts and Strogatz. Some common properties of complex networks, such as Internet servers, power grids, human communities, and disordered porous media, are mainly determined by the way of connections among their vertices or nodes.

According to Watts and Strogatz, a small-world network is characterized by a clustering coefficient C and a path length L. The clustering coefficient measures the probability that given node a is connected to nodes b and c then also nodes b and c are connected. The shortest path length from node a to node b is the minimal number of connections need to get from a to b. A small-world network is defined by two properties.

a) First, the average clustering coefficient C is larger than for a corresponding random network with the same number of connections and nodes. The clustering coefficient expected for a regular rectangular grid network is zero while for a random network the probability p of connection of two nodes is the same for neighboring nodes as for distant nodes.

b) Second, the average path length L scales like logN, where N is the number of nodes. For regular grid networks L scales as N^d, where d is the dimension of the space and for random networks L scales as logN. As result, C is large and L is small in a small-world network.

Brain neural networks study has great interest from several points of view, the brain and its structural features can be seen as a prototype of a physical system capable of highly complex and adaptable patterns in connectivity, selectively improved through evolution, architectural organization of brain cortex is one of the key features of how brain system evolves, adapts itself to the experience, and to possible injuries. According to many research literatures, brain neural network is a small-world network, which also has some properties as small-world network. So we can use a small-world network simulate the neural network of human.

4.1.3 Comparison between SWN and ANN

The differences of ANN and small-world network consist in three complex levels: vertices, links and behavior.

a) The first level of complexity is denoted by diversity of vertices. The diversity of vertices in ANN is also showed by the three different facets: number, type and threshold. But the diversity of vertices in small-world network is showed only by their number. There is no means about vertex in a small-world network, but the vertex has a threshold in ANN. At the same time, there are many vertices in a small-world network, and it is far more than that of ANN.

b) The second level of complexity is represented by diversity of links[32,33]. The diversity of links in ANN is also showed by the three different facets: number, weight and distribution. The diversity of links in small-world network is presented by the two different facets: number and distribution. There are a few links in ANN which have different weight. Contrarily, there are many links in small-world network which have no weight. On the other hand, the distribution of links is asymmetric in small-world network, but it is regular in ANN. The small-world

network has a large cluster coefficient and a shortest average path because of the asymmetric distribution of links.

c) The third level of complexity is represented by diversity of their behavior. The ANN can simulate some ability of brain neural network, but it is not too accurate but approximate. The small-world network can describe many real complex systems including the brain neural network, but it cannot used to simulate the function of brain, such as memory, association, etc. The contrast of artificial neural network and small-world network is shown in table 5. Small-world networks are highly clustered networks with small distances among the nodes. There are many biological neural networks that present this kind of connections. There are no special weightings in the connections of most existing small-world network models. However, this kind of simply-connected models cannot characterize biological neural networks, in which there are different weights in synaptic connections.

Table 5. The contrast between artificial neural network and small-world network

Complex level	**Artificial neural network (ANN)**	**Small world network (BNN)**
Vertex	Diversity of number, type and threshold	Diversity of number
Link	Diversity of number, weight and distribution	Diversity of number and distribution
Behavior	Diversity of behavior	Diversity of behavior

Chapter 3 PROBLEM STATEMENT

An optical MIN can be implemented with either free-space optics or guided wave technology. It uses the Time Division Multiplexing. To exploit the huge optical bandwidth of fiber, the Wavelength Division Multiplexing (WDM) technique can also be used. With WDM the optical spectrum is divided into many different logical channels, and each channel corresponds to a unique wavelength. Optical switching involves the switching of optical signals, rather than electronic signals as in conventional electronic systems. Two types of guided wave optical switching systems can be used.

1. The first is a hybrid approach in which optical signals are switched, but the switches are electronically controlled .With this approach, the use of electronic control signals means that the routing will be carried out electronically. As such the speed of the electronic switch control signals can be much less than the bit rate of the optical signals being switched. So, with this approach there is a big speed mismatch occurs due to the high speed of optical signals.
2. The second approach is all-optical switching. This has removed the problem that occurred with the hybrid approach. But, such systems will not become practical in the future and hence only hybrid optical MINs are considered.

Problems in Optical Networks are:

1. Path Dependent Loss
2. Optical crosstalk

Two ways are used to solve crosstalk problem in Optical Networks:

1. Space Domain Approach
2. Time Domain Approach

In this thesis neural network solution is used in order to avoid crosstalk in OMINs and various routing algorithms have been analysed such as :

1. Shortest path Algorithm
2. Routing in Three Stage Interconnection Networks
3. Routing Algorithm using ANN

Once the OMIN is chosen, the routing neural network can be constructed and the weights never have to change. The usefulness of this routing method depends on the speed of the Hopfield neural network as well as other requirements of the system. The routing performance using neural network approach is better as compared to routing in OMINs that do not have self-routing capabilities. In this thesis, to solve the optical traffic routing problems in communication networks, neural based computational algorithm have been used. Once the shortest path is obtained it can carry the entire traffic for a given OD pair, provide its capacity is not exceeded. This thesis also includes a number of Neural Network Applications such as:

1. Switching
2. Object Recognition
3. Call Routing
4. Network Design of a Banyan Network Controller
5. A Neural Network Model for Traffic Controls in MINs
6. Parallel Neural Networks for Speech Recognition

Chapter 4 ROUTING ALGORITHMS

1. Shortest Path Algorithm

The main function performed by a routing algorithm is the selection of routes for various origin-destination(OD) pairs. There are two main performance that are substantially affected by the routing algorithm, the throughput (quantity of service) and the average delay (quality of service). A good routing algorithm should select the routes which have minimum average delay. In the shortest path algorithm, a cost is associated with every link in the network. In most cases, the cost is proportional to the delays. The objective is to find a multilink path joining two nodes that has minimum total cost. Different implementations of the shortest paths algorithm, in both synchronous and asynchronous fashion. The neural network structure of the algorithm was first introduced by Ruch and Winarske. Their method. however, has serious limitations. It can find the shortest path for a given OD pair only when the number of links that the path contains is known, which is an unrealistic assumption [15,20,37].

1.1 Neural Network Computation Algorithm

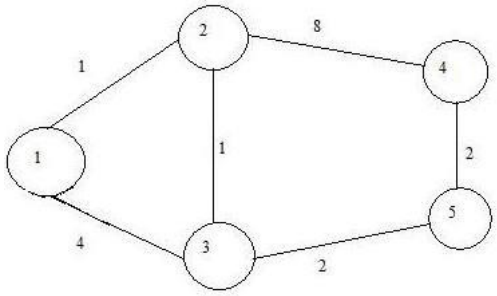


Figure 21. A 5-Node Network[1]

Rauch and Winarske[3] suggested that the solution of the shortest path algorithm can be represented by a 2-dimensional neuron array V = (V_{ij}) with each output of the neuron in

the array having value V_{ij} = 0 or 1. The number of rows in the array is equal to N, the number of nodes in the network, and the number of columns is equal to the number of nodes that the path contains.

Table 6. The shortest path connecting node 1 and node 5 for the 5-node network

	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	0
5	0	0	0	1

It is obvious that for the array to represent a valid path, there can be only one nonzero entry in each column and there can be at most one nonzero entry in each row (this condition is different from the one required by the TSP problem). An nonzero entry in the ij^{th} position of the array can be interpreted as node i is the jth node in path. Using this representation, a total NxM neurons are needed to represent all the paths having length (number of nodes in the path) M. Given an OD pair, the first and the last column of the array are fixed, so there are Nx(M-2) active neurons in the amy which are free to be updated.

The problem with this representation is that if we do not know how many nodes the shortest path would contain, i.e. M is unknown. To overcome this limitation of the R-W method, we fix the number of columns in the array at N, which is the maximum possible number of nodes any path could contain in an N-node network.

2. Routing in Three Stage Interconnection Networks

A routing representation for routes in a MIN will now be constructed. The problem of routing a call through a three stage interconnection network requires the choice of an appropriate route through the network. A neural network solution refers to a specific

neural network architecture and set of interconnection weights that result in a neural network capable of solving a specific problem [37]. Even as calls are routed or removed from the interconnection network only the external neural network inputs change not the interconnection weights [37,47]. The neural network solution applies to any three stage interconnection network e.g. a three stage Clos network. A three stage interconnection network consists of an input , a middle and an output stage. In order to route a call, which is a desired connection from an input line of an input section to an output line of an output section, a center section with an available link to both the appropriate input section and the appropriate output section must be locate [46,47].

2.1 Example: Traveling Salesperson Problem [46]

The neural network solution presented here is an adaptation of that used by Hopfield and Tank [19] for the traveling salesperson problem (TSP). Given a list of cities and their geographic locations, the traveling salesperson problem is to choose the shortest path for visiting every city once and only once. The path must start and end in the same city. Hopfield and Tank's approach for the traveling salesperson problem involves choosing an energy function where local minima should correspond to valid short tours. Their solution uses a square matrix of neurons to represent the tour, such that a firing neuron in the (i,j)th location indicates visiting the ith city during the jth time interval. For the tour to be valid there must be exactly one firing neuron in each column and exactly one firing neuron in each row. An additional constraint on the tour length is added to favor shorter tours.

3. Routing Algorithm using ANN[2,40,49,57]

A routing representation will be in a state of minimal energy when the neuron outputs directly represent a legal routing array. Figure 23 shows two routes , namely a 2-12 route and a 13–16 route. Only one of the many possible routing solutions for these two desired connections is shown. Each message route will have a corresponding routing matrix [41]. For example, the routing matrix for the 2-12 message in figure 23 is shown as the

leftmost table in table 7. The columns of a routing matrix represent the output ports for each stage of the interconnection network.

If ($a_{i,j}$) = 1

the message is routed through outport port i of stage j.

If ($a_{i,j}$) = 0

the message is not routed through outport port i of satge j.

Thus, 2-12 message is routed through output port 4 of stage 1 , output port 15 of stage 2 , and outport port 12 of stage 3. The routing matrix for the 13-16 message is shown as the rightmost table in table 7. The routing array for the set of messages is simply constructed by treating each routing matrix as a slice and constructing a loaf. The routing array is a 3 – dimensional representation of a set of routes, and each slice of the array represents a single route. For our example , there are two messages to be routed so the routing array will have two slices. In general, if a system has m input ports, there can be m slices in the routing array.

Each element of the routing array now has three indices :

If element $a_{i,j,k} = 1$

then message i is routed through output port k of stage j.

We say :

If i = l and k = n

$a_{i,j,k}$ and $a_{l,m,n}$ are in same row

If i = l and j = m

They are in same column

If j = m and k = n

Finally they are in same rod.

A legal routing array will satisfy following three constraints

a) One and only one element in each column is equal to 1.
b) The elements in successive columns that are equal to 1 represent output ports that can be connected in the interconnection network.
c) No more than one element in each rod is equal to 1. Each neuron in ANN directly represents an element in the routing array for an interconnection network and message set.

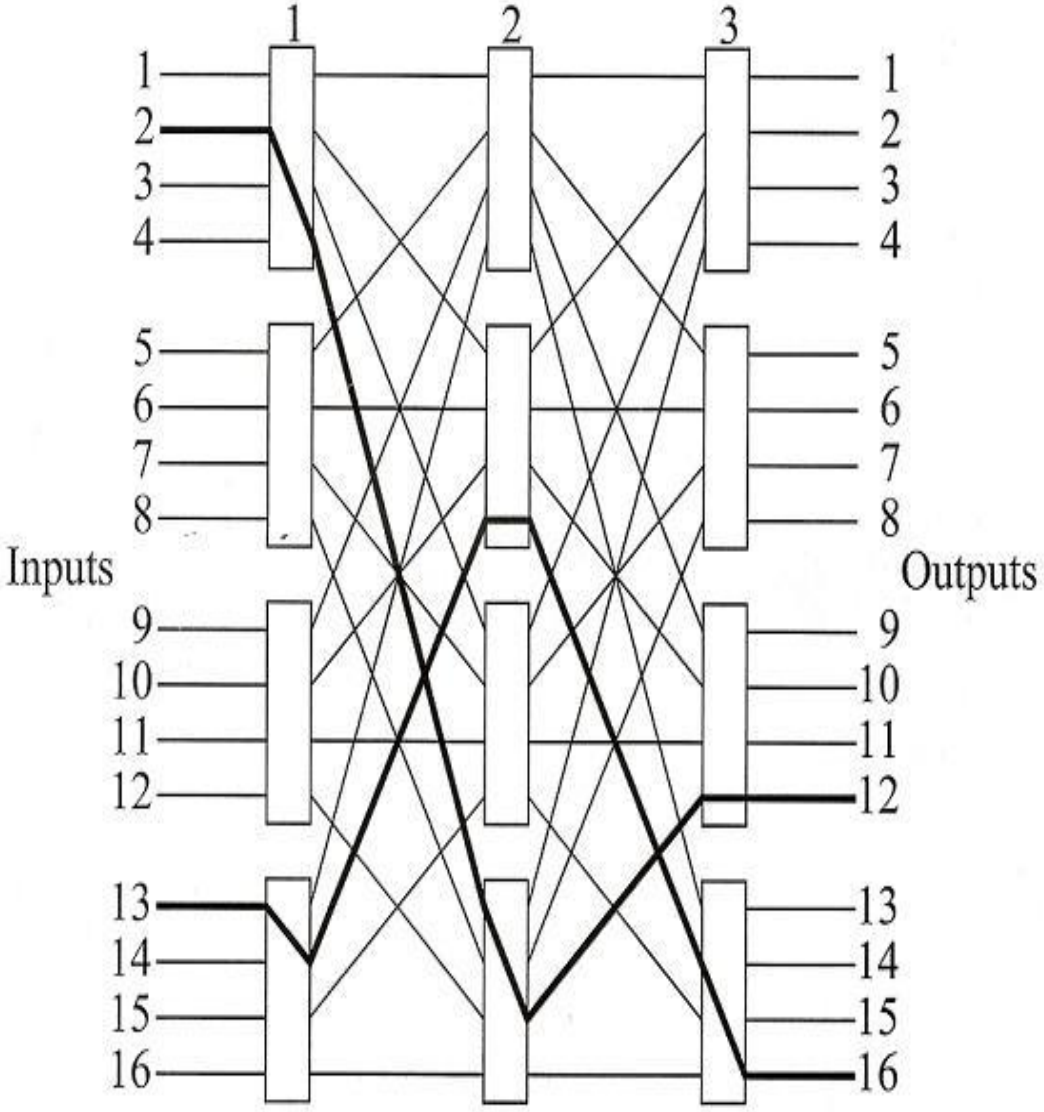


Figure 22. Interconnection network having 2-12 and 13-16 connections[44]

Table 7. Matrix representation for the 2-12 and 3-13 connections

Stage

Outport port	1	2	3
1	0	0	0
2	0	0	0
3	0	0	0
4	1	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	1
13	0	0	0
14	0	0	0
15	0	1	0
16	0	0	0

Stage

Outport port	1	2	3
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	1	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	1	0	0
15	0	0	0
16	0	0	0

Chapter 5 APPLICATIONS OF NEURAL NETWORKS

1. Medical[56]

ANNs are currently a hot research area in medicine and it is believed that they will receive extensive application to biomedical systems in the next few years. At the moment, the research is mostly on modelling parts of the human body and recognising diseases from various scans (e.g. cardiograms, CAT scans, ultrasonic scans, etc.). Neural networks are ideal in recognising diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so the details of how to recognise the disease are not needed. What is needed is a set of examples that are representative of all the variations of the disease. The quantity of examples is not as important as the quantity. The examples need to be selected very carefully if the system is to perform reliably and efficiently.

1.1 Modelling 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. 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 analysed. In medical modelling 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.

1.2 Electronic noses

ANNs are used experimentally to implement electronic noses. Electronic noses have several potential applications in telemedicine. Telemedicine is the practice of medicine over long distances via a communication link. The electronic nose would identify odours in the remote surgical environment. These identified odours would then be electronically transmitted to another site where an door generation system would recreate them. Because the sense of smell can be an important sense to the surgeon, telesmell would enhance telepresent surgery.

2. Switching[50,55]

Switching Neural Networks are a class of systems that have many simple processors-neurons-which are highly interconnected. The function of each neuron is simple, and the behavior is determined predominately by the set of interconnections. Thus, a neural network is a special form of parallel computer. Although a major impetus for using neural networks is that they may be able to learn the solution to the problem that they are to solve, argue that another, perhaps even stronger, impetus is that they provide a framework for designing massively parallel machines. The highly interconnected architecture of switching networks suggests similarities to neural networks. Abroad Band integrated services digital network(B-ISDN)consists of two major parts: optical transmission with synchronous optical networks (SONET) and switching with an

asynchronous transfer mode (ATM) [2] switch. Although today's optical terminal can transport signals at several gigabits per second, the ATM switching fabric can function at rates up to only a few hundred megabits per second. The broad-band network research community has spent much time and effort on the design of ATM switch fabrics. One approach in designing ATM switch fabrics is to use interconnection networks [3] that were originally designed for a multiprocessor's parallel and distributed processing. There are several different interconnection networks, e.g. Banyan, baseline, shuffle exchange, and delta. Among them, the Banyan network is the most popular to be used as a basic building block in ATM switch fabric designs.

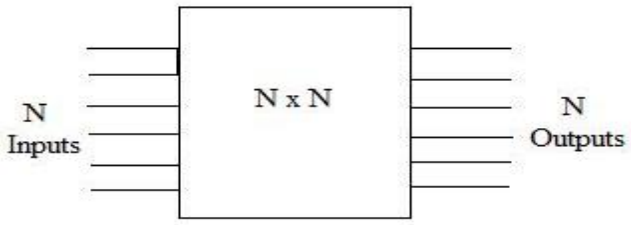


Figure 23. N x N Crossbar Switch Concept[55]

3. Building a Cheaper Artificial Brain[59]

It has been the dream of the first author for more than a decade to build artificial brains, which he defines as being interconnected sets of evolved neural network circuit modules. The basic approach is to use evolutionary engineering to evolve the individual modules, each with its own simple function, and hence fitness definition. After evolution, these modules are downloaded into the RAM memory of a PC, and interconnected according to the artificial brain (A-Brain) designs of human BAs (Brain Architects). However, the bottleneck with this approach has always been the very slow speeds of the neural net module evolution. Typically, with a PC, it can take hours to over a day to evolve a single module. Obviously, if one is to build an A-Brain, consisting of 10,000s of modules or more, this slow evolution time is impractical and unacceptable. Somehow, the evolution speeds need to be accelerated significantly. Two major attempts have been made by the first author and his teams to solve this problem. The first attempt used a very expensive

piece of specialized hardware called a CAM Brain Machine (CBM) that cost $500,000 each. The CBM was developed during the years 1997-2001. The second attempt, described in this application, is far cheaper, at around $3000. It has been made possible due largely to the phenomenon of Moore's Law, i.e. the doubling of electronic capacities of chips every year or so. It is now possible to build FPGA chips that contain literally millions of programmable logic gates. The second attempts uses an off the shelf commercial electronic board with an FPGA of 3-10 million logic gates. This is now sufficient to evolve neural net circuit modules at electronic speeds with a single chip. The price of such boards is about $1500. The price and capacity of such boards will probably cause a revolution in brain building, making it far cheaper, and hence affordable by any university or commercial research lab interested in building artificial brains.

4. Evolving the Neural Network Model for Forecasting Air Pollution Time Series[19]

The forecasting of air quality is one of the topics of air quality research today due to urban air pollution and specifically pollution episodes i.e. high pollutant concentrations causing adverse health effects and even premature deaths among sensitive groups such as asthmatics and elderly people. A wide variety of operational warning systems based on empirical, causal, statistical and hybrid models have been developed in order to start preventive action before and during episodes. In recent years, the considerable progress has been in the developing of neural network (NN) models for air quality forecasting.

5. Object Recognition[13,31]

While artificial neural networks are typically robust enough that many different topologies can be used to learn the same set of data, the topology chosen still impacts the amount of time required to learn the data and the accuracy of the network in classifying new data. Thus, choosing a good topology becomes a crucial task to the success of the neural network. One approach is that of combining a neural network with a genetic algorithm. The genetic algorithm would create population of potential structures for the

neural network[7]. The neural network would then briefly try each of the structures and report on the success of each. Using these values from the neural network, the genetic algorithm would then evolve a new population for the network to try. After several generations, a population of several good structures should result and could then be used to completely train the neural network. Using data from successfully experiments in identifying objects, a new effort was undertaken to see if the concept of combining a genetic algorithm with the neural network could produce results as good or better than the already successful network. ANNs have been used successfully in a large number of applications. They have been used to effectively recognize handwritten or spoken words, allowing for dictation software and software which can take handwritten text and convert it to electronic text. They have been used to distinguish between the handwriting of different individuals, providing a method for handwriting verification. GAs are capable of searching through a large number of potential solutions to find good solutions. There have been numerous approaches that combine neural networks with genetic algorithms. These generally fall into three categories of:

a) using GA to determine the structure of an NN
b) using GA to calculate the weights of an NN
c) using GA to determine both the structure and the weights of an NN.

a) In the first category, GA is used in an attempt to find a good structure for a neural network. The process involves the GA evolving several structures and using the neural network as the fitness function to determine the fitness level of each structure.
b) In the second category, the genetic algorithm is used to evolve the weights of the network instead of using back propagation or some other technique for determining the weights. This can potentially result in quicker training of the network.
c) GA is used to generate not only the structure, but also the connection weights in the GA. For each structure in the population, another GA evolves weights. Using

the evolved weights, the GA sets a fitness value for that network structure. Thus, two GAs are implemented with one running inside the other.

6. Call Routing[52]

The aim of this application was to investigate the possibility of using Hopfield nets to find optimal call routes. Essentially this is equivalent to using a Hopfield network to solve the travelling salesman, optimisation problem. To use the technique on a particular problem it is necessary to find an appropriate set of parameters for the network. Despite considerable investigation we were unable to find an acceptable set, and this project was abandoned.

7. Neural Network design of a Banyan Network Controller[57]

Packet switching networks require queueing due to output blocking that is unavoidable[11]. If internal blocking is also present, such as with the Banyan switching network, then longer queues can be expected. The algorithm for choosing nonblocking sets of data cells from the queues can significantly affect the throughput and queueing behavior. Using a novel equivalence class approach, this algorithm can be reduced on a Banyan network to a constraint satisfaction problem. To gain the required computational speed, massive parallelism of neural networks is used. A neural network design is defined using multiple overlapping winner-take-all circuits. This is shown to be stable and only result in nonblocking sets of data cells. An efficient interface between the neural network and the queue is also defined. The performance of the Banyan with a neural network controller is compared to a noninternal-blocking switch[9] with various controllers. Surprisingly, the Banyan is within a factor of two of the nonblocking switch.

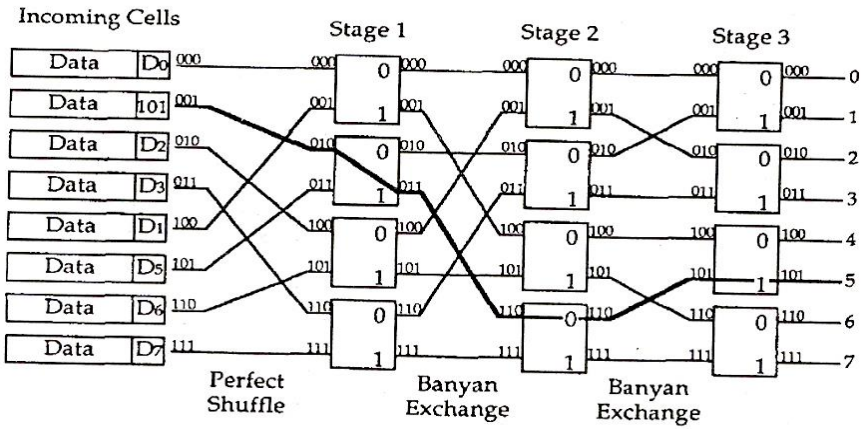


Figure 24. An 8x8 Banyan switching network with self routing[9]

8. A Neural Network Model for Traffic Controls in Multistage Interconnection Networks[20]

The goal of the neural network model is to find conflict-free traffic flows to be transmitted among given I/O traffic demands in order to maximize the network throughput[3]. The model requires n^2 processing elements for the traffic control in an nxn multistage interconnection network. The model runs not only on a sequential machine but also on a parallel machine with maximally n^2 processors. The model is verified by solving ten 32x32 network problems.

9. Parallel Neural Networks for Speech Recognition[7]

Backpropagation algorithm[9] works successfully in most cases for a multi-layered feedforward neural network. But because of a gradient descent method to reduce errors, it can be stuck in local minima. Many approaches require significant admounts of computation[4,8] to reduce the chance of local minima. One of feasible way is to provide more hidden units. However, overfitting problems may arise by adding more parameters and it may affect the generalization ability. Two main approaches are tried to find an optimal size of the neural network. First, it starts with a minimal neural network and adds new hidden units one by one [3]**.** Second, it starts from a large number of redundant

weights of a multi-layered neural network and delete weights selectively through second-derivative information[5].

For increasing learning accuracy, multiple learned classifiers has been attractive recently [1,2,10]. Our approach is a kind of a cascaded neural network. Without heavy computation, our approach employs several neural networks with different weights, each of which handles a different subset of the problem. Each neural network explores different search space and selection strategies are applied to decide the final result as opposed to a single neural network that explores all search spaces and handles all problems. Each neural network processes only those instances which it can handle and passes the other instances to subsequent neural networks. Many studies using neural networks have been done in the area of speech recognition [7]. Our approach to a speech recognition problem is demonstrated.

CONCLUSIONS

A neural network routing methodology was presented that is capable of providing control bits to OMINs. This routing method is valid for a wide range of OMINs that have a certain structure. Once the OMIN is chosen, the routing neural network can be constructed and the weights never have to change. For a new message set, only certain boundary bias currents need to be varied. The usefulness of this routing method depends on the speed of the Hopfield Neural Network as well as other requirements of the system. The fact that a Hopfield Neural Network can be readily constructed in an optical computing environment makes the neural network routing approach quite attractive for OMIN routing problems. The routing performance degrades as the MIN size increases and as the number of messages in a message cycle increases. Depending on the implementation of the neural network router, the routing methods described in this thesis may have advantages over many other routing methods in terms of speed.

In this thesis neural based computational algorithm has been used for solving optimal traffic routing problems in communication networks. The key idea in this algorithm is the introduction of pseudo links which allow the extension of any path to a length N path so that it can be represented by an NxN neuron array. Once the shortest path is obtained by using this algorithm, it can carry the entire traffic for a given OD (origin-destination) pair, provided its capacity is not exceeded.

This thesis also discussed a NN solution capable of routing calls though a three stage interconnection network. This solution used fixed interconnection weights for the NN and uses fewer neurons. Traveling salesperson problem is to choose the shortest path for visiting every city once and only once. The path must start and end in the same city. The Banyan switching network discussed in this thesis is a popular architecture due to its simplicity, and potential for high data rates.

FUTURE SCOPE

1. The Neural Network can be designed to avoid crosstalk in Optical irregular MINs.
2. The Neural Network approach can be used to design routing algorithm for multipath OMINs.
3. The Neural approach can also be used for fault tolerant MINs.
4. Neural Network can be used for OMINs that do not have self-routing capabilities.

REFERENCES

[1] A.A. Sawchuck, B.K. Jenkins, C.S. Raghavendra, and A. Varma, "Optical crossbar networks," IEEE Computers, vol. 20, no. 6, pp. 50-60, June 1987.

[2] A.J. David and B.E. Saleh, "Optical implementation of the Hopfield algorithm using correlations," Applied Optics, vol. 24, no. 9, pp. 1469-1475, 1985.

[3] Beahrs OH, Henson DE, Hutter RVP and Kennedy BJ, "Manual for staging of cancer," 4th edition, Addison-Wesley, 1992.

[4] Bhuyan, L.N., Agrawal, and D.P., "Design and performance of generalized interconnection networks," IEEE Transaction Computers, vol.32, no.2, pp. 1081–1090, 1983.

[5] Bhuyan, L.N., Yang, Q., Agrawal, and D.P, "Performance of multiprocessor interconnection networks," IEEE Computers, vol. 22, no. 2, pp. 25–37, 1989.

[6] Blaket James T. and Trivedi Kishor S., "Reliabilities of Two Fault-Tolerant Interconnection Networks," IEEE Computers, vol. 18, no. 5, pp. 300-305, 1988.

[7] Byoung Jik Lee , "Parallel Neural Networks for Speech recognition," IEEE Computers, vol. 5, no. 6, pp. 2093-2097, 1997.

[8] B. Potter, J. Sinclair, and D. Till., "An Introduction to Formal Specifications and Z," 3rd edition, Prentice Hall, 1991.

[9] B. S. Yandell, "Practical Data Analysis for Designed Experiments," 3rd edition, Chapman & Hall, 1997.

[10] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," IEEE Computers, vol. 23, no. 3, pp. 119-130, May 1988.

[11] Cam Hasan, "Rearrangeability of (2n − 1)-Stage Shuffle-Exchange Networks," Society for Industrial and Applied Mathematics, vol. 32, no. 3, pp. 557-585, 2003.

[12] C. Brodley and P. Smyth, "The process of applying machine learning algorithms," Proceedings of Workshop on Applying Machine Learning in Practice at IMLC-95, vol. 17, no. 4, pp. 0018-0033, 1995.

[13] Christopher M. Taylor and Arvin Agah, "Evolving Neural Network Topologies for Object Recognition," World Automation Congress (WAC), vol. 8, no. 5, pp. 2048-2054, July 24-26, 2006.

[14] C. W. Johnson, "Using Z to support the design of interactive safety-critical systems," IEE/BCS Software Engineering Journal, vol. 16, no. 2, pp.49–60, March 1995.

[15] D. Bersekas and R. Gallager, "Data Networks," 2nd edition, Prentice-Hall, 1987.

[16] D. Rodvold., " A software development process model for artificial neural networks in critical applications," IEEE Computers, vol. 9, no. 4, pp. 3317-3323, July 1999.

[17] D.T. Sannella and A. Tarlecki, "Extended ml: An institution- independent framework for formal program development," Category Theory and Computer Programming Journal, vol. 40, pp. 364-389, September 1985.

[18] Fahlman SE and Lebiere C, "The cascade correlation leaming architecture," National Science Foundation contract EET-8716324, Carnegie Mellon University, vol. 8, no. 6, pp. 3204-3218, 1991.

[19] Harri Niska, Teri Hiltunena, Ari Karppinenb, Juhani Ruuskanena and Mikko Kolehmainen , "Evolving the neural network model for forecasting air pollution time series," Engineering Applications of Artificial Intelligence, vol. 4, no. 17, pp. 159–167, 2004.

[20] H.E. Rauch and T. Winarske, "Neural networks for routing communication traffic," IEEE Control Systems Magazine, vol. 18, no. 2, pp.26-31, April 1988.

[21] H. van Vliet., "Software Engineering," 3rd edition, Wiley, 2000.

[22] I. Sommerville, " Software Engineering," 3rd edition, Addison-Wesley, 1989.

[23] J. A. McDermid, "Formal methods: Use and relevance for the development of safety critical systems" In P. A. Bennett, editor, Safety Aspects of Computer Control. Butterworth- Heinemann, Oxford, UK, vol. 9, no. 6, pp. 1024-1032, Sept. 1993.

[24] J. Jacky, " Specifying a safety-critical control system in Z," IEEE Transactions on Software Engineering, vol. 18, no. 2, pp. 99–106, Feb. 1995.

[25] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. Natl. Acad. Sci. USA, vol. 79, pp. 2554-2558, April 1982.

[26] J. J. Hopfield, and D. W. Tank, "Computing with Neural Circuits: A Model," IEEE Computers, vol. 33, pp. 625-633, August 1986.

[27] J. J. Torres, M. A. Munoz, J. Marro, and P. L. Garrido, "Influence of topology on the performance of a neural network," Neuro computing, vol. 49, no.3, pp. 229-234, 2004.

[28] J. P. Bowen and V. Stavridou, " Safety-critical systems, formal methods and standards," IEE/BCS Software Engineering Journal, vol. 8, no. 4, pp.189–209, July 1993.

[29] J. Spivey, "The Z Notation: A Reference Manual," 2nd edition, Prentice Hall, New York, 1992.

[30] Leon Sterling ,Anthony Senyard and Ed Kazmierczak, "Software Engineering Methods for Neural Networks" Proceedings of the Tenth Asia-Pacific Software Engineering Conference(APSEC'03), vol. 4, no. 5, pp. 1362-1530, 2003.

[31] Liang Chen, "Pattern classification by assembling small neural networks," Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, vol. 3, no. 5, pp. 3309-3315, July 31 - August 4, 2005.

[32] Luis G. Morelli, Guillermo Abramson, and Marcelo N. Kuperman, "Associative memory on a small-world neural network" Eur. Phys. J. B, vol. 38, no. 5, pp. 495-500, 2004.

[33] LI Shou-wei, "Analysis of Contrasting Neural Network with Small-world Network," International Seminar on Future Information Technology and Management Engineering, vol. 2, no. 3, pp. 1098-1113, 2008.

[34] L. M. Barroca and J. A. McDermidF, "Formal methods: Use and relevance for the development of safety-critical systems" The Computer Journal, vol. 35, no. 6,pp. 579–599, Dec. 1992.

[35] L. Smith, "A framework for neural net specification," IEEE Transactions on Software Engineering, vol. 18, no. 7, pp. 601-612, 1992.

[36] L. Smith, "Using a framework to specify a network of temporal neurons," Technical report, University of Stirling, vol. 6, no.9, pp. 607-621, 1996.

[37] L. Zhang and S.C.A. Thomopoulos, "Neural network implementation of the shortest path algorithm networks," in Proceedings of the International Joint Conference on Neural Networks, vol. 34, no. 4, p.591-601, June 1989.

[38] M. D. and M. R., "Hierarchical evolution of neural networks," Proceedings of the 1998 IEEE Conference on Evolutionary Computation, vol. 45, no.8, pp. 908-915, 1998.

[39] M. E. J. Newman, "The structure and function of complex networks," SIAM Review, vol. 45, no. 7, pp. 167-256, 2003.

[40] M. Goudreau and C. Giles, "Routing in random multistage interconnection networks: Comparing exhaustive search, greedy and neural network approaches," International Journal of Neural Systems, vol. 3, no. 2, pp. 234-245, 1992.

[41] M. Goudreau and C. Giles, "Routing in optical multistage interconnection networks: a Neural Network Solution," published in IEEE/OSA Journal of Lightware Technology, vol. 13, no.6, pp. 1111-1118, 1995.

[42] M. P. and M. R. , "Culling and teaching in neuro-evolution," Proceedings of the 7th International Conference on Genetic Algorithms, vol. 7, no. 4, pp. 768-775, 1997.

[43] M. Winikoff, P.Dart and E. Kazmierczak, "Rapid prototyping using formal specifications," Proceedings of the Australasian Computer Science Conference, vol. 34, no. 5, pp. 1239-1246, Feb 1998.

[44] M.W. Goudreau and C.L. Giles, "Neural network routing for random multistage interconnection networks," Advances in Neural Information Processing Systems 4, Morgan Kaufmann Publishers, vol. 7, no.9, pp. 772-729, 1992.

[45] P. J. Sallis, G. Tate and S. G. MacDonell, "Software Engineering," 2nd edition, Addison-Wesley, 1995.

[46] P.J.W. Melsa, J.B. Kenney and C.E. Rohrs, "A neural network solution for routing in three stage interconnection networks," Proceedings of the 1990

International Symposium on circuits and Systems, vol.46, no. 7, pp. 483-486, May 1990.

[47] P.J.W. Melsa, J.B. Kenney and C.E. Rohrs, "A neural network solution for call routing with preferential call placement," Proceedings of the 1990 Global Telecommunications Conference, vol. 23, no. 4, pp. 1377-1382, December 1990.

[48] P. Meseguer and A. Preece, "Assessing the role of formal specifications in verification and validation of knowledge based systems," Proceedings of the Third International Conference on Achieving Quality in Software, Edited by S. Bologna and G. Bucci, London, Chapman & Hall, vol. 32, no.6, pp. 317–328, 1996.

[49] P. M. Grant and J. P. Sage, "A Comparison of Neural Network and Matched Filter Processing for Detecting Lines in Images," in J. S. Denker AIP Conference Proceedings 151, Neural Networks for Computing, Snowbird Utah, AIP, vol. 22, no. 4, pp. 342-355, 1986.

[50] Regis Bates, "Optical Switching and Networking Handbook," 2nd edition, McGraw-Hill, New York, 2001.

[51] R. Golden, "A unified framework for connectionist systems," Biological Cybernetics, vol. 41, no. 20, pp. 109-115, 1988.

[52] R.J. Frank, S.P. Hunt and N. Davey, "Applications of Neural Networks to Telecommunications Systems," IEEE Transaction computers, vol. 7, no. 5, pp. 987-990, 1882.

[53] R. Pressman, " Software Engineering - A Practitioner's Approach," 3rd edition, McGraw Hill, 1992.

[54] R.P.Lippmann, "An introduction to computing with Neural Nets," IEEE ASSP Magazine, vol. 34, no. 7, pp. 4-22, April 1987.

[55] T. Brown, "Neural Networks for switching," in IEEE Communications Magazine, vol. 27, no. 11, pp. 73-81, November 1989.

[56] T. L. Huston, A. E. Smith, and J. M. Twomey, "Artificial Neural Networks as an Aid to Medical Decision Making: Comparing a Statistical Resampling Technique with the Train-and-Test Technique for Validation of Sparse Data Sets,"

Proceedings of AI in Medicine: Interpreting Clinical Data, Stanford University, vol. 14, no. 3, pp. 1304-1310, 1994.

[57] T.X. Brown and K.H. Liu, "Neural Network Design of a Banyan Network controller, " IEEE Journal on Selected Areas of Communication, vol. 18, no. 8, pp. 1428-1438, October 1990.

[58] W. W. Royce, "Managing the Development of Large Software Systems," Proceedings of IEEE WESCON, vol. 44, no. 2, pp. 2333-2339, 1970.

[59] Wang Ce, Thayne Batty and Hugo de Garis, "Building a Cheaper Artificial Brain," Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, vol. 13, no. 7, pp. 674-683, July 31 - August 4, 2005.

[60] http://www.informatik.unistuttgart.de/ipvr/bv/projekte/Neuroinformatik/model.html

PAPERS PUBLISHED

1. Kiranpreet Kaur and Rinkle Aggarwal, "A Survey of Self Organising Networks," National Conference on Impact of IT on Society – Emerging Trends & Issues, DAV College Amritsar, Punjab, February 28 - March 01, 2009.

2. Kiranpreet Kaur and Rinkle Aggarwal, "Routing in Optical Network using Neural Networks," National Conference on Advances in Computer Networks & Information Technology, Guru Jambheshwar University of Science & Technology, Hisar, Haryana, pp. 99-104, March 24 – 25, 2009.