

Most current artificial neural networks exist only within software simulators running on conventional computers. Simulators can provide great flexibility, but require immensely powerful and costly hardware for even very small networks.

An artificial neural network implemented as a custom integrated circuit could operate many thousands of times faster than any simulator as each neuron can operate simultaneously. A significant problem with implementing neural networks in hardware is that larger networks require a great deal of silicon area, making them too costly to design and produce.

In this book, I test the effectiveness of a number of algorithms that reduce the size of a trained neural network while maintaining accuracy.

*Published by Fultus Corporation  
www.fultus.com*

Michael Oldroyd is a software development veteran who started programming professionally back in 1992. He is now development manager at AES Data Systems.

He has worked as a consultant and software developer for a number of international organisations including Mobil Oil, The European Commission, Deutsche Bank, Compaq Computer, and the Cabinet Office.

He has developed several bespoke AI trading and decision support tools used on trading floors in the currency, stock and energy markets.

He is a professional member of the IEEE and the Computational Intelligence Society.



Optimisation of Massively Parallel Neural Networks • Michael Oldroyd

# Optimisation of Massively Parallel Neural Networks

Michael Oldroyd

---



# Optimisation of Massively Parallel Neural Networks

by

**Michael Oldroyd**

ISBN 1-59682-010-1

Copyright © 2004 by Michael Oldroyd

All rights reserved.

Cover Art by Ethan Bailey




Published by Fultus Publishing

Publisher Web Site: [www.fultus.com](http://www.fultus.com)

Fultus eLibrary: [elibrary.fultus.com](http://elibrary.fultus.com)

Online Book Superstore: [store.fultus.com](http://store.fultus.com)



No part of this book may be used or reproduced in any manner whatsoever without written permission except in the case of brief quotations embodied in reviews and critical articles.

The author and publisher have made every effort in the preparation of this book to ensure the accuracy of the information. However, the information contained in this book is offered without warranty, either express or implied. Neither the author nor the publisher nor any dealer or distributor will be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

# Table of Contents

Bibliography .....	9
Introduction.....	11
Getting to Hardware .....	11
Massive Parallelism.....	12
Fault Tolerance .....	12
Small Ones Are More Juicy .....	12
Chapter 1. Overview.....	15
1.1 Abstract .....	15
1.2 Fault Tolerance .....	15
1.3 The Biological Neuron .....	16
1.4 The Artificial Neuron.....	17
1.5 Overview of Connectionist Model .....	17
1.5.1 The Synapse .....	18
1.5.2 Summation .....	19
1.5.3 Thresholding.....	19
Chapter 2. Limitations .....	21
2.1 Abstract.....	21
2.2 The significance of connections.....	21
2.3 No Optimisation.....	21
2.4 Connect Adjacent Layers Only.....	22
2.4.1 The Unit Square.....	23
2.4.2 The Two Spirals Problem .....	25
2.5 Local Connections.....	27
2.5.1 Using Neurons as Consolidators .....	29
2.5.2 The Wolf Detector .....	31
Chapter 3. Reduction Algorithms.....	35
3.1 Abstract.....	35

3.2	Connection Removing Methods .....	35
3.2.1	Remove Connections at Random .....	35
3.2.2	Remove connection where a neuron has > X inputs.....	36
3.2.3	Remove connection where the weight of a connection is < X% of total input weights	36
3.2.4	Remove connection where the weight of an input connection is less than X% of the neuron threshold.....	36
3.2.5	Remove connection where the number of inputs > X% of the layer width .....	36
3.2.6	Remove connection where the number of inputs > X% of the number of neurons .....	37
3.2.7	Remove connection where the weight is of smallest magnitude .....	37
3.3	Eliminating redundant neurons .....	37
3.4	Compensation for removal .....	37
3.5	Recurrent Networks .....	39
3.5.1	The Jordan Model .....	40
3.6	Network Evaluation .....	42
3.6.1	Feedforward Propagation.....	42
3.6.2	Parallel Relaxation.....	42
Chapter 4	Software Development .....	45
4.1	Abstract .....	45
4.2	Component Parts of the Project .....	45
4.2.1	Generating a network .....	46
4.2.2	Optimising the network.....	46
4.2.3	Testing the Optimised Network .....	46
4.3	Analysis and Design .....	47
4.3.1	Methodology .....	47
4.4	Data Definitions .....	48
4.4.1	Network File Format .....	48
4.5	Network Generator .....	49
4.6	Simulator / Optimiser .....	50
4.7	Utility Functions .....	51
4.8	File Functions.....	52
4.9	Graphic Functions.....	53
4.10	Network Manipulation.....	56
Chapter 5	Results and Discussion.....	63
5.1	Testing Strategy .....	63

5.2	Combinatorial Explosion .....	63
5.3	Data Output.....	63
5.3.1	Relative number of connections.....	63
5.3.2	Relative number of neurons .....	63
5.3.3	Output units accuracy.....	64
5.3.4	Output patterns accuracy .....	64
5.3.5	Unit Accuracy Ratio.....	64
5.3.6	Pattern Accuracy Ratio.....	64
5.4	Test Criteria .....	64
5.5	Number of Layers.....	65
5.6	Width of Layers.....	66
5.7	Threshold Compensation .....	69
5.8	Multiple Algorithm Applications .....	72
5.9	Different Algorithms .....	76
<b>Chapter 6. Conclusions and Future Work.....</b>		<b>79</b>
6.1	Conclusions.....	79
6.1.1	Tolerance .....	79
6.1.2	Algorithms .....	79
6.1.3	Compensation.....	79
6.1.4	Strange Attractors .....	79
6.2	Future Work.....	80
6.2.1	Larger Networks .....	80
6.2.2	Actual Networks .....	80
6.2.3	Actual Inputs .....	80
6.2.4	Parallel Relaxation.....	81
6.2.5	Grouping of outputs .....	81
<b>Appendix A. Program Code.....</b>		<b>83</b>
	Generate.CPP – ANN Generator.....	83
	Optimise.CPP – ANN Optimiser and Simulator .....	92
<b>Appendix B. Test Data .....</b>		<b>137</b>
	Simplest.NET.....	137
	Loop.NET .....	141

# Table of Figures

Figure 1.1 – A Biological Neuron .....	16
Figure 1.2 – An Artificial Neuron.....	17
Figure 1.3 – Rosenblatt's Perceptron.....	18
Figure 2.1 – A Layered Network.....	22
Figure 2.2 – Unit Square for logical AND.....	24
Figure 2.3 – Unit Square and network for logical XOR.....	25
Figure 2.4 – The Two Spirals Problem .....	26
Figure 2.5 – The Spiral Problem solved using a layered network.....	27
Figure 2.6 – In this network neurons only connect to adjacent layers. ....	28
Figure 2.7 – Two and three input neurons .....	29
Figure 2.8 – The correct two input equivalent of the three input network.....	31
Figure 2.9 – The Wolf Detector made using a maximum of 3 input connections.....	32
Figure 3.1 – Threshold Compensation Example .....	38
Figure 3.2 – Distribution of Total Inputs for Compensation Example.....	39
Figure 3.3 – A recurrent Jordan network .....	41
Figure 3.4 – Four Stable States of a Particular Hopfield Network.....	43
Figure 4.1 – Top Level Data Flow Diagram .....	46
Figure 4.2 – Structure Chart of Network Generator .....	50
Figure 4.3 – Overview of Project Functions .....	51
Figure 4.4 – Structure chart for getdata .....	52
Figure 4.5 – Structure Chart for drawnet.....	54
Figure 4.6 – Sample Output from drawnet function.....	55
Figure 4.7 – Sample output from drawneuron function.....	56
Figure 4.8 – Structure chart for findloop function .....	57
Figure 4.9 – LOOP.NET showing state units in black .....	58
Figure 4.10 .....	59
Figure 4.11 – Search tree from neuron 8.....	60
Figure 4.12 – Search tree from neuron 11 .....	60

Figure 4.13 – Structure chart for removeconn .....	61
Figure 4.14 – Test network before neuron 7 is removed.....	62
Figure 4.15 – Test network after neuron "7" has been removed .....	62
Figure 5.1 .....	65
Figure 5.2 .....	66
Figure 5.3 .....	67
Figure 5.4 .....	68
Figure 5.5 .....	70
Figure 5.6 .....	71
Figure 5.7 – After One Application .....	72
Figure 5.8 – After Two Applications .....	73
Figure 5.9 – After Three Applications .....	73
Figure 5.10 – After Four Applications.....	74
Figure 5.11 – After Five Applications.....	74
Figure 5.12 .....	75
Figure 5.13 .....	76
Figure 5.14 .....	77
Figure 5.15 .....	78

# Bibliography

Artificial Intelligence (2nd Edition)

- Elaine Rich, Kevin Knight

Perceptrons (Expanded Edition)

- Marvin L Minsky, Seymour A Papert

What's Hidden in the Hidden Layers

- Klaus J Obermeier, Janet J Barron

Building Blocks for Speech

- Alex Waibel, John Hampshire

Back Propagation

- William P Jones, Josiah Hoskins

Experiments in Artificial Neural Networks

- Ed Rieteman

# Introduction

Neural networks have moved from being simply software simulations in university laboratories. They now form an integral part of many real world systems that affect our everyday lives.

- In government, Neural networks are used by Transport for London, to read the number plates of every vehicle entering and exiting London's Congestion Charging Zone.
- In finance, neural networks are used by many financial institutions including JXX capital in New York who use a neural network based system to predict hedge fund prices, using the worlds largest commercial supercomputer.
- In film, the Oxford based company "NaturalMotion", uses neural networks to create lifelike behaviour in 3D animated models with minimal human interaction. Their software brought to life vast armies of virtual soldiers in Peter Jackson's final Lord of the Rings film, "Return of the King" and Wolfgang Petersen's "Troy".
- In Security, Neural networks in chip form are embedded in fingerprint recognition systems, retinal scanners and many other biometric devices.

## Getting to Hardware

Most current neural networks, exist only within simulation programs running on a conventional computer. Simulating the function of a neural network in this way provides a great deal of flexibility, and of course allows manufacturers and software providers to leverage their investment in existing technology.

Networks created, trained and executed using a simulator are capable in principle of exactly the same tasks as a network created using dedicated electronics, but they fail to take advantage of two key strengths that set neural networks apart from conventional computers, namely massive parallelism and fault tolerance.

---

## Introduction

---

### Massive Parallelism

With a conventional Von Neumann computer, only one instruction is executed per processor at once. Even the largest conventional computers have a relatively small numbers of processors. At time of writing, the HP9000 superdome (the most powerful server available from HP) can be fitted with a maximum of 128 processors, hence 128 simultaneous operations.

Because a neuron can operate independantly of other neurons around it, each neuron in a neural network can become a separate processing unit. If the neural network is implemented as a dedicated chip, each of the five to six thousand neurons in a typical neural network could run simultaneously. For large manufacturing runs using VLSI, a single chip could contain millions of neurons.

Add to this the huge efficiency advantage of implementing each neuron as a few electronic gates, without all the overhead of a microprocessor, operating system and simulation program and it seems quite apparent that a dedicated chip with a unit cost of a few pennies could be orders of magnitude quicker than even the fastest supercomputers.

### Fault Tolerance

If you open up your computer, find a component on the motherboard, however small, and lever it out with a pair of pliers, then try and switch it all back on, there is an extremely good chance that you'll just be looking at a blank screen. It won't nearly work, or be nearly OK but just get some sums a bit wrong occasionally. it simply won't respond at all. Except in a very limited sense, conventional computers are simply not fault tolerant.

If you take a neural network, and remove some neurons, or cut some of the wires that connect the neurons, then the quality of decisions that the neural network comes to will be affected, but not disasterously. Because all of the processing and memory is distributed around the system, losing any small part, will not cause immediate failure.

### Small Ones Are More Juicy

The larger the number of neurons and connections in a neural network, the greater the complexity of the tasks that can be performed, but also the greater the silicon area required to implement it in chip form. There is often a fixed limit to the number of neurons and connections that can be physically fitted onto a chip. This limit is much less than the numbers possible using a software simulator.

From nature, we know that a lot can be achieved with only a small neural net. A bumble bee manages to get by with only about one thousand neurons.

In this book, I discuss and quantify the effectiveness of a number of algorithms for maximising the accuracy of a trained neural network given a reduced number of neurons and connections.

# Chapter 1.

## Overview

### 1.1 Abstract

In our quest to produce truly intelligent machines we have one naturally occurring example of what is possible, the biological brain. While we are unable to imitate the simplest of animal brains, a rather promising line of research is that of producing artificial brains using the same building blocks that occur in nature.

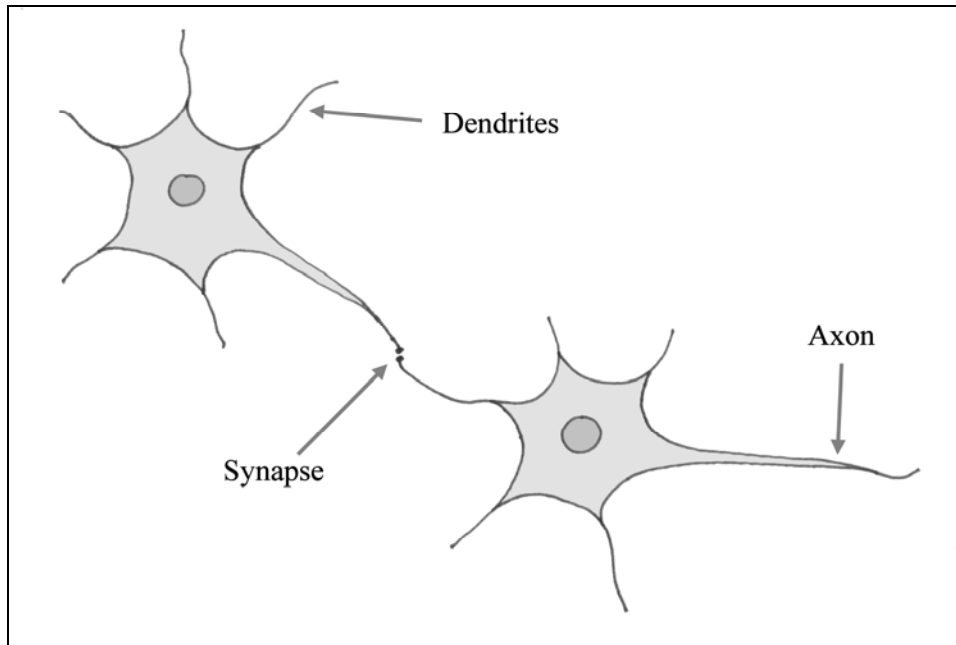
### 1.2 Fault Tolerance

By the time we are about 12 years old, we have all the brain cells we shall ever have (approximately one hundred billion). After that time, the brain cells start to die, yet humans tend to function, without apparently suffering because of this loss, for many decades. It appears that a naturally occurring neural network can function perfectly adequately with fewer neurons than it started with.

In 1984, J J Hopfield showed that a network model made up of inter-connecting cells could be stimulated to produce a finite set of output patterns. He also showed that the same output patterns could be produced even after some of the units were removed.

In this book I attempt quantify the effects of removing elements of a neural network, under differing conditions.

### 1.3 The Biological Neuron



**Figure 1.1 – A Biological Neuron**

A natural brain is built up of a large number of simple processing elements, or neurons. Electrical signals originate in the sensory organs, then pass through dendrites into the neuron.

Depending on the signals received, the neuron may send a single electrical output signal along its axon. The axon is connected to the dendrites of other neurons via a synapse. Some Neurons (motor neurons) are connected to glands and muscles to trigger action in response to the input from the senses.

## 1.4 The Artificial Neuron

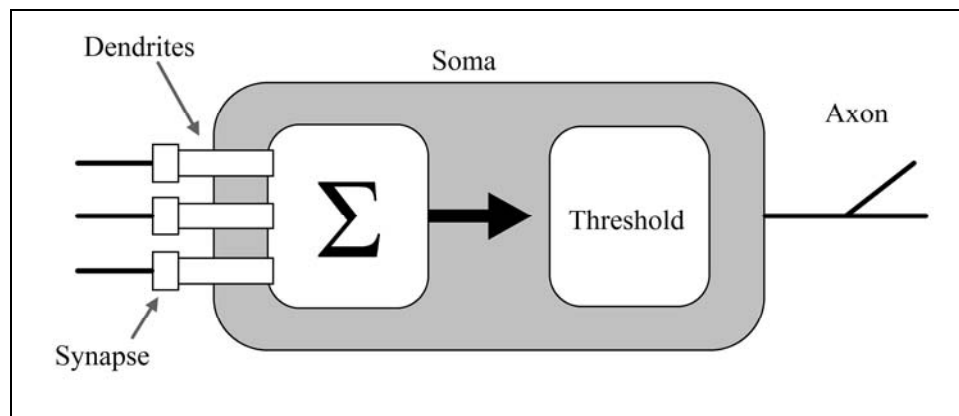


Figure 1.2 – An Artificial Neuron

Recent advances in computing and neural sciences have opened up the possibility of greatly increasing the computational efficiency of previously very difficult tasks by several orders of magnitude. Today a neural network based computer is considered the most promising 5th generation computing machine.

While each unit performs a very simple process at what can be quite a low speed (a natural electrochemical neuron operates much more slowly than an artificial electronic one), each neuron within a layer can process different data elements and the rich connectivity between neurons allows data elements to be combined and computed by different sections of the network.

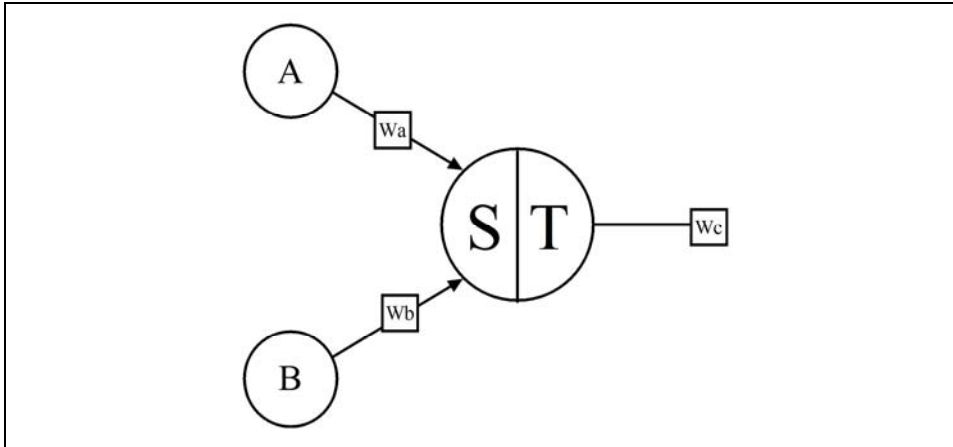
This allows a very high degree of distributed processing around the network, which effectively eliminates the computational bottlenecks associated with the traditional Von Neumann architecture.

This book recounts an investigation of methods of reducing the size of neural networks, by reducing the degree of connectivity between neurons.

## 1.5 Overview of Connectionist Model

A Connectionist model is an attempt to mimic a very powerful natural computer, the human brain. A Neural Network is a representation of the working of a natural brain. There

are many models for a neural network. One of the more widely known is based the concept of the perceptron, an invention of Frank Rosenblatt, described in his book, "Principles of Neurodynamics" published 1962. An illustration of Rosenblatt's Perceptron is shown.



**Figure 1.3 – Rosenblatt's Perceptron**

Logically the perceptron model corresponds closely to the biological neuron as described above. The summation and threshold functions are both present and given by the S and T sections of the perceptron body respectively. The synapses correspond to the weighting functions  $W_a$  and  $W_b$ . In the above diagram A and B represent other perceptrons.

Inputs come from A and B. These are weighted by values  $W_a$  and  $W_b$  to give the weighted inputs  $I_a$  and  $I_b$ . These inputs are then added together and if the total is above the threshold value, the perceptron fires.

A signal undergoes three different processes when passing through a neural net.

### 1.5.1 The Synapse

Before entering a neuron, the signal undergoes a weighting function. In a biological system this function is performed by the synapses, the dendrites carrying the weighted result. The weight of a particular connections determines the degree of importance of a particular signal in relation to other signals entering the neuron. In most models, the weights are represented by real numbers between -1 and +1 and the value of the incoming signal is multiplied by the weight before entering the neuron. Note that weights can be positive or

negative thus the magnitude of the weight affects the significance of the individual input while the sign of the weight determines whether a high input will incite or inhibit the firing of the neuron. The weights are altered during the learning process to give the network its characteristics.

### 1.5.2 Summation

All the weighted signals entering a neuron are added together.

### 1.5.3 Thresholding

The summed values are put through an adjustable threshold processor. The threshold value for a particular neuron, is also altered during the learning process. In some cases the threshold is simply implemented as another weight. In this case the threshold determines the likelihood of a particular neuron to fire, irrespective of its inputs. In a biological system, when the potential inside the cell body is larger than the threshold value, the cell outputs a single electrical signal along the axon (or fires). Note that the amplitude of the output is not proportional to the inputs along the dendrites. The neuron either fires at full power, or not at all.