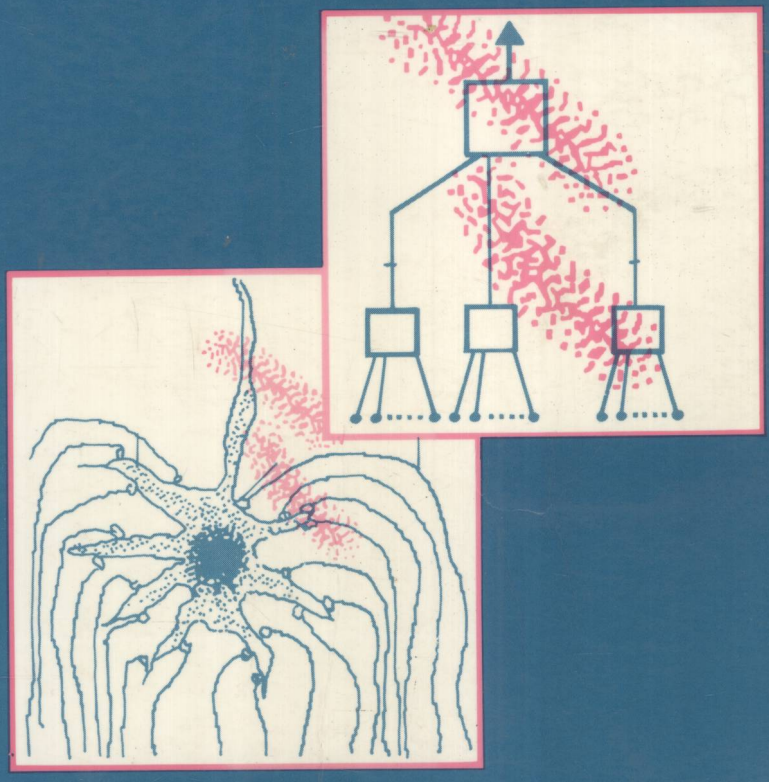
An Introduction to

NEURAL COMPUTING



Igor Aleksander and Helen Morton

CHAPMAN AND HALL

An Introduction to Neural Computing

Igor Aleksander
Professor of Neural Systems
Head of Department of Electrical Engineering
Imperial College
London

and

Helen Morton
Lecturer
Department of Human Sciences
Brunel University
Middlesex

CHAPMAN AND HALL
LONDON • NEW YORK • TOKYO • MELBOURNE • MADRAS

UK	Chapman and Hall, 11 New Fetter Lane, London EC4P 4EE
USA	Chapman and Hall, 29 West 35th Street, New York NY10001
JAPAN	Chapman and Hall Japan, Thomson Publishing Japan, Hirakawacho Nemoto Building, 7F, 1-7-11 Hirakawa-cho, Chiyoda-ku, Tokyo 102
AUSTRALIA	Chapman and Hall Australia, Thomas Nelson Australia, 480 La Trobe Street, PO Box 4725, Melbourne 3000
INDIA	Chapman and Hall India, R. Sheshadri, 32 Second Main Road, CIT East, Madras 600 035

First edition

Typeset in Great Britain by
KEYTEC, Bridport, Dorset
Printed in Great Britain by
T. J. Press (Padstow) Ltd, Padstow, Cornwall

ISBN 0 412 37780 2

British Library Cataloguing in Publication Data

Aleksander, Igor, 1937–
An introduction to neural computing.
1. Artificial intelligence
I. Title II. Morton, Helen
006.3

ISBN 0-412-37780-2

Library of Congress Cataloging-in-Publication Data available

An Introduction to Neural Computing

Remembering Bettina
1912 – 1989

A software package simulating many of the neural networks described in this book is available from Unistat Ltd, PO Box 383, Highgate, London N6 5UP and in North America from Adhoc Reading Systems Inc, 28 Brunswick Woods Dr, East Brunswick, NJ 08816, USA. Tel: 201-254 7300, Fax: 201-254 7310, who will be happy to supply further information and quote prices for supplying the software on floppy disk and issuing site licences for its use.

Introduction
Why the fuss?

The late 1980s will be remembered among computer scientists as the time when an area of study called *connectionism* or *neural networks* or *parallel distributed processing* suddenly came to the fore not only as a topic for research but also as an area for commercial development. All the above names are synonyms: they refer to machines that, unlike conventional computers, have a structure that, at some level, reflects what is known of the structure of the brain.

As we shall show, this is not entirely a new field: indeed its past stretches back beyond that of conventional computing. What is new, however, is a concern with well-founded analysis and a deepening understanding.

In this introduction we briefly review some of the history of the connectionist way of doing things and give reasons for the revival of interest and, consequently, the reasons for writing this book. Ultimately the rationale for connectionism has to do with the fact that mechanisms do matter and, in order to design machines that work well, mechanisms need to be understood. We also preview the plan for the rest of the book: simple aspects of neural system analysis and design.

IS THERE A FUSS?

In 1987 the American Institute of Electrical and Electronic Engineering called the first conference on neural nets. It took place at San Diego, California where 200 authors presented their papers to 2000 delegates. It was described as 'the dawn of a new era'; the scientists were talking of a kind of computing that is inspired by the cellular networks of living brains. The following year saw even bigger events: more papers and more delegates. So, as far as most computer scientists were concerned, something new was going on.

The next two years saw a considerable influx of research funding, which at the time of writing, is still on the increase. Major learned societies have been set up world-wide, and the subject has now been

embraced by most research centres in computing and electrical engineering and by some centres in the natural, life and human sciences.

The subject has an interdisciplinary flavour not known in technology since the 1960s when 'cybernetics' (the study of information and control in man and machine) held some researchers' attention. Neural computing, however, has fired the imagination of a larger community. It is normal at conferences on neural computing for as many contributions to come from physicists, biologists, neuropsychologists and statisticians as come from computer scientists. This reflects the character of this science: it defines an interdisciplinary culture which is based on brain-like learning, as opposed to traditional computing which is based on programming. The fuss is about the discovery of a new form of computing which is distinct from the traditional variety, and which seems likely to open up a host of new possibilities.

THE CONVENTIONAL COMPUTING CULTURE: ALGORITHMS

Computers have become so much a part of our lives that we forget that their structure is based on just one person's view of the way in which a computational process can be organized. We are referring to John von Neumann's incredibly far-sighted view of the way a computer worked. The computing is done by a single arithmetic/logic unit that operates on data held in memory. A series of instructions, also held in memory, controls where the data is obtained by this unit, what is to be done with it, and where the result is to be directed in the memory (von Neumann, 1947). No matter how sophisticated the task performed by a computer, the machinery is eventually called upon to execute long lists of elementary instructions. The strength of the technique lies in the scope for packing groups of instructions together to create more elaborate instructions. For example, a computer might execute the instruction to multiply two numbers together as a series of simpler, addition instructions. $A \times B$ means adding A to itself B times. Computer languages such as Pascal or PROLOG consist of so-called 'high-level' instructions which are made up of complexes of simpler instructions. Loading such a language into a computer involves providing the machine with a 'compiler' or an 'interpreter' for that language. These are series of instructions that translate the high-level commands into low-level ones that the machine can execute. (A compiler does the translation on complete programs, while an interpreter does it line-by-line.)

Artificial intelligence (AI) makes use of the highest levels of this packaging in order to get a computer to do things which if done by humans would be said to require intelligence. For example, a computer that plays chess merely executes a massive series of instructions, such as 'if the board state is A, then find all the possible next moves which can

be made from state A . . .'. The rules for doing this are arranged in an ordered series of levels. For example, the above instruction is at a 'high' level, and needs to be broken up into simpler steps at lower levels. At the next level down are rules which test a given move, to see whether it is applicable to that particular board state or not. At an even lower level there are rules that determine the legality of the applicable rules within the conventions of the game of chess, and so on.

The concept that is used at each of these levels is the *algorithm*. An algorithm is a 'series of steps that achieves a desired aim'. Multiplying A by B may require the algorithm: 'to achieve $A \times B$ add A to itself B times.' Similarly, to make a chess-playing program work, the computer needs algorithms such as: 'for board state A, test for all applicable rules' and: 'for each applicable rule and for each board piece to which it applies, apply the rule and store the resulting board state.' To choose the best move, the program must also be given a way of evaluating the listed board states: this too is a series of algorithms.

The central question here is, where do these algorithms come from? The answer is simple: they are representations of human knowledge. The humble repetition of additions to achieve multiplication is a representation of the basic arithmetic that we learn as children at school. The algorithms that go into an AI chess-playing or problem-solving program are representations of human knowledge of the rules and strategies for chess or for solving problems. The art of the programmer is that of turning algorithms into code that the computer can understand through the use of a suitable computer language, while the art of the AI scientist is to find algorithms for the forms of computer behaviour he or she is trying to achieve. Complicated hierarchies and interactions between algorithms are what make a computer work, whether it is crunching numbers or playing chess. All of these algorithms must be invented and implemented by a human being to give a computer a semblance of intelligence.

The sobering implication of this is that the range of things that can be done with a conventional computer is limited to those tasks for which a human can find algorithms. This simple fact distinguishes information processing in conventional computers from human information processing: humans are capable of developing their behaviour through 'learning' while computers have to wait for some human to feed them the algorithms required to accomplish the desired task.

THE NEURAL COMPUTING CULTURE: EXPERIENCE ACQUISITION

While living creatures are endowed with some predispositions when they are born, they mostly become viable through a process of gathering 'experience' about the environment in which they live. This occurs

through a process of exploration or through interaction with other living creatures, for humans have developed sophisticated languages whereby they can interchange experience. However, the point which is being stressed is that the difference between this style of becoming a competent 'mechanism' and the way in which a computer acquires its competence by being programmed is quite enormous.

Programming a computer involves spelling out to it every step of a process. A human being relates his or her conversations to experience. A child who is told 'be gentle with the new kitten' may or may not obey the command, depending on a whole variety of previous events that the child may have experienced. The child may not know what the word 'gentle' means; he or she may have been scratched by a cat and not wish to approach the kitten at all; or he or she may have previously been rough with a cat which then hissed and ran away. The way that a kitten responds to a child may also be determined by its previous experience of children. So, stored experience is the basis from which the interaction with the world is interpreted by living creatures.

In many ways, if one's aim is to build machines that work well, doing it by conventional programming seems a more direct and controlled way of doing things. Experience seems to be too closely related to the makeup of an individual and, if this mode were to exist in machines, it might lead to individualistic and enigmatic devices. Their control might become expensive and unreliable. But neural computing is not about building humanoid machines: it is about discovering how machines might store and use experience, should this be necessary to the design of devices with skills that cannot be achieved by programming. Are there such skills?

WHERE EXPERIENCE SCORES OVER PROGRAMMING

The period between the mid-1960s and the mid-1980s was a time of great euphoria and confidence among those working in artificial intelligence laboratories. The style of step-by-step design of programming algorithms seemed set to conquer a vast variety of tasks which 'if done by humans would be said to require intelligence' (as defined earlier). Programming rules were being discovered that could allow a machine to look for good moves at chess rather than exhaustively looking at all possible moves; general methods of solving clearly stated problems were being developed; robot planning programs were shown to be feasible and algorithms were being developed that would extract meaning from sentences input into the computer in something that resembled natural language.

The first warning signs were sounded in the UK in 1973 in a review by Sir James Lighthill (Lighthill, 1973). He pointed out that most of these methods relied on getting the computer to sift through vast amounts of data and that the techniques were being tested on highly restricted 'toy-like' problems. He foresaw that if the methods were to be applied to real problems the power and speed of computers would have to grow at a rate which was greater than that which was conceivable: he called this problem the 'combinatorial explosion'. It is this graphic notion of the computer having slavishly to search through myriads of possibilities that begins to point to the limitations of the programming approach. When a person sees an old friend he or she does not search a database of stored images of all the people he or she knows. The recognition, if it happens at all, is almost instantaneous. So the brain does not succumb to the combinatorial explosion. In any case, a search through stored images may well be a totally fruitless algorithm. The face might be older, distorted by a grimace or partly obscured by hair – how then does one effect a match with similar faces stored in memory?

In recent years, much work in artificial intelligence has been tidied up under the heading of 'expert systems'. An expert system is an algorithmic method for trying to capture experience by using two standard components: a memorized list of rules (such as: 'if X is a child of A and Y is a child of A then X and Y are siblings') and ways of inferring conclusions from these rules and some supplied facts in order to answer a question. For example, the facts might be: 'Mary is the child of Joan' and: 'Fred is the child of Jack': using the rule quoted earlier, the machine would say 'no' to the question: 'are Mary and Fred siblings?' unless there is another rule that deals with the possibility of Joan and Jack being married.

With reference to vision tasks such as face recognition, the key point is that there are no obvious rule sets that can equal the performance of the brain in important tasks such as recognizing a friend, or reacting rapidly to a dangerous situation while driving. A typical example of a task that has unfortunately proved a stumbling block for the application of expert systems is the provision of guidance to unmanned vehicles from the image 'seen' by a computer through a television camera. Such vehicles might be important in the exploration of dangerous or remote areas. Is the image in fig. 0.1(a) a road or a tree? Notice how once little clues are added in fig. 0.1(b) and 0.1(c), it is easier to decide that 0.1(b) is a tree, and 0.1(c) is a road. Of course, rules could be developed to distinguish between these images, but the occurrence of each slightly different distinguishing clue would require the addition of yet another rule. Thus a seemingly endless explosion of rules seems to be inevitable whenever the algorithmic approach is applied to this type of input,

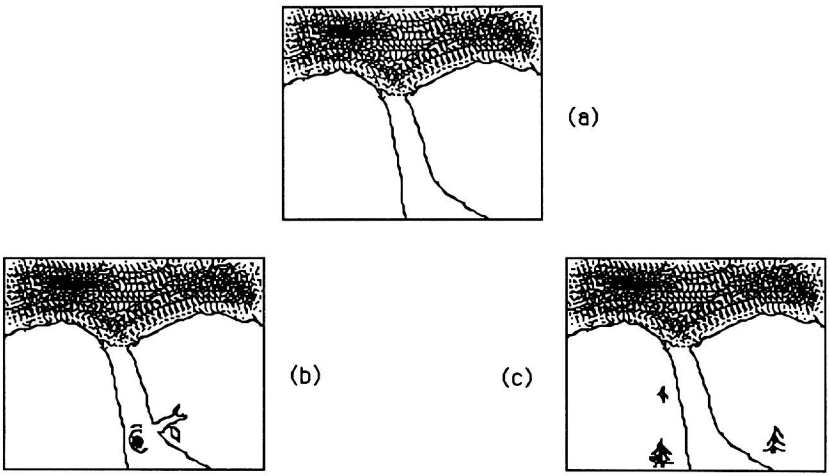


Fig. 0.1 (a) Tree or road? (b) clue for tree (c) clue for road.

making the understanding of images and speech the veritable Achilles' heel of algorithmic, rule-based methods.

Human beings find these tasks easy, and all they have to go on is their experience stored in a network of interconnected brain cells (neurons). The 'computing' that these devices perform is clearly different in kind from the algorithmic, rule-based method. So the quest for investigators of neural nets is to identify how it is that these networks are capable of storing and using experience.

TURING, OTHERS AND BRAINS

Many will be aware of the fact that in 1936 Alan Turing, a British mathematician, laid down the principles for defining what is and what is not computable. He reduced a computation machine to a reading head armed with simple rules for reading symbols on a tape, printing new symbols, and moving to another point in the tape. These rules, which Turing showed would exist for any task that could be said to be computable, are the first manifestations of the concept of an algorithm. But not so many will be aware of the fact that Turing was highly conscious of the fact that brains do their computations in ways that do not depend on algorithms (Turing, 1936). He wrote of the way in which the brain did its computations through cycles of activity in neural nets. He rejected this as a way of thinking about computing machines solely because it did not help with theorizing about computable numbers, which was the subject that motivated his personal interest in theoretical computing machines at that time.

Indeed, John von Neumann, another mathematician, and a giant in the history of the design of computing machines and hence a contributor to the algorithmic way of computing (von Neumann, 1947), also talked of neural nets, seeing them as providing a way of doing computations that emerges from the structure of the net itself. Notable is his concept of 'self-reproducing automata' (Burks and von Neumann, 1966), where neural structures are analysed for their ability to transfer a behaviour that takes place in one part of a net to another part of the net.

Working at the Massachusetts Institute of Technology, Norbert Wiener, whose name is associated with the definition of cybernetics, saw the interplay of logic and network structure as being fundamental to a mathematical understanding of computation in both brains and machines (Wiener, 1947).

Therefore it seems that the algorithmic method has always been seen as the well-behaved, predictable kind of computing, while the neural mode has always been seen as the less predictable, but nonetheless powerful, way of carrying out computations. Indeed, it has always been recognized that neural networks, unlike conventional computers, have properties that emerge from the structure of the hardware, a topic that is only now being properly discussed.

So how is it that the algorithmic methodology has shot ahead so fast, and neural work lain largely dormant for so long? The rest of this book traces the development of neural methodology in a chronological fashion so as to show why there was a collapse of interest in the late 1960s followed by a revival in the 1980s. In Chapter 1 the basic principle is explained in a general way how a neural net does its computations. We show that a net can retrieve stored knowledge by reconstructing learnt patterns of activity from partial clues. This is the 'autoassociative' mode. It is contrasted with the 'associative' style in which a net learns to associate an output with a given input and to produce roughly the right output even if the input is slightly distorted. These two properties are human-like and lead to high hopes that such systems may extend artificial intelligence out of the domain limited by discoverable rules, and allow for fast retrieval of data and the better understanding of living brains.

YEARS OF DEFINITION: 1943–1969

The starting point for most who have studied neural networks has been a model of the fundamental cell of the living brain: the neuron. The recognized US pioneers who first suggested such a model were the neurophysiologist Warren McCulloch and the logician Walter Pitts. In 1943, much in the spirit of cybernetics, they developed a simple model of variable resistors and summing amplifiers that represents the variable

synaptic connections or weights which link neurons together and the operation of the neuron body (soma), respectively. In Chapter 2 the behaviour of this highly influential model is analysed fully, which was not only adopted by the pioneers of neural computing but also provides the basis of most nets that are being discussed today. Another important idea was put forward in 1949 when the seeds of a model of dynamic memory were sown by a neuropsychologist, Donald Hebb, who suggested that a group of neurons could reverberate in different patterns, each of these patterns relating to the recall of a different experience (Hebb, 1949). In Chapter 1 (section 1.5) it is shown how the learning mechanisms embedded in simplified models control these groups of reverberating neurons.

Probably the 'buzz-word' that was most heard in cybernetic circles in the 1960s was 'perceptron'. It was coined in 1962 by Frank Rosenblatt who used the word to refer to a system which recognized images using the McCulloch and Pitts model in conjunction with some non-learning 'feature extractors' of an image. To some extent this modelled what was known of the early stages of primate vision (Rosenblatt, 1962). This is another influential model that is very much part of contemporary discussion, and Chapter 2 deals with those elements of perceptron design that have survived to the present day.

ANALYSIS AND IMPLEMENTATION

The early work done in the mid-1960s led to an understanding of the capabilities of the nets in so far as they could be simulated on contemporary computers. But this empirical approach was severely challenged in 1969 in a seminal book written by Marvin Minsky and Seymour Papert (Minsky and Papert, 1969). In Chapter 3 the nature of their objection is explained which lies in the inability of some neural nets to learn to perform tasks that require 'internal representations' not explicitly specified by the training data. For example, a neural net that has learnt to distinguish between images of cars and bicycles may have, within it, a net which has learnt to recognize the presence of two wheels, and another which recognizes four wheels. These are what one calls internal or 'hidden' representations. Because the net must somehow work out these representations for itself, training turns out to be a difficult task. This issue is very much part of current debate and most modern learning algorithms and net structures are being designed with the possibility of solving this difficulty in mind.

Once most of the theoretical basis has been put in perspective, one can then ask how a neural computer could ever be made. In Chapter 4 the issues involved in implementing neural nets are introduced. Most nets are currently being studied as programs running on serial computers

or on special fast cards that plug into conventional computers. But the future of this subject turns upon the ability of engineers to make systems which are physically parallel, and in Chapter 4 we consider the questions that arise from the implementation of neurons as silicon chips: should these be analog, digital, mixtures of the two or something different altogether? Should machines have the ability to learn or should the learning be off-line, with the neural net coming into play only when it is required to solve a problem? We also introduce the notion that random access memories (RAMs) might be used to model neurons. Such memory devices have been developed and perfected for conventional computers and have the advantage that they are already being mass produced and are thus readily available for use in neural nets. In a very broad principle, they perform the function of the neuron – they vary their behaviour on demand – but it is important to face the central question of what is lost and gained by this approach.

In Chapter 5, with the above in mind, we describe the WISARD (**WI**kie, **S**tonham's and **A**leksander's **R**ecognition **D**evice), first built in 1981, which demonstrated that a large machine based on neural principles could be built. The device learns to recognize patterns of about 250 000 picture points in 1/25th of a second. The principles of this machine and ways of understanding its performance are discussed.

THE REVIVAL

There is little doubt that it was John Hopfield of the California Institute of Technology who was responsible for a revival of interest in the analysis of neural nets (Hopfield, 1982). But what was the basis of this breakthrough? Interestingly, his orientation was very practical: he saw the stable reverberations mentioned above as being a good way of making advanced computer memories. However, in retrospect, it was his analysis of such systems, which was closely related to the analysis of systems of physical particles, that impressed the scientific community. This is the subject of Chapter 6.

But Hopfield's analysis did not tackle the problems of 'hard learning' which still cast a large shadow over the future of neural nets as devices that could actually be used to carry out useful computations. Many also realized that the net could get stuck in the wrong reverberations. It was Geoffrey Hinton (then at Carnegie Mellon University in Pittsburgh) who suggested a way of overcoming these problems through what he called 'the Boltzmann machine', named after the Austrian physicist Ludwig Boltzmann (Hinton, Sejnowski and Ackley, 1984). Boltzmann added much to physics by studying the effect of heat on the agitation of molecules. In Chapter 7 it will be shown that it is the similarity between systems of physical particles and neural nets which enabled Hinton to

use the electronic equivalent of heat (which is electronic 'noise') to avoid false reverberations and deal with the hard learning problem.

No sooner had they defined the Boltzmann machine, than Hinton and his colleagues started asking important questions about whether there was a real need for a close analogy with physical systems. They concluded that 'feed-forward' networks which do not reverberate required investigation (Rumelhard, Hinton and Williams, 1986). These serve to take a pattern at the input of the net and feed it forward to the output where it is translated into another pattern – this can lead to the association of input patterns, such as a written word, with output patterns, which might control a voice synthesizer so that it utters the appropriately related sounds. We discuss this technique in Chapter 8.

MATHEMATICS

The work of Hopfield, Hinton and others who have followed in their wake is grounded in mathematics. We have not avoided the use of mathematical notation altogether. This is not only because it provides a useful and compact language for representing the principles that govern the behaviour of nets, but also because anyone wishing to delve more deeply into the literature of the subject will need to be able to decipher this mode of expression.

We have tried to tread a narrow path between too much and too little mathematics. So that the reader should not be put off by such notation, we have provided some guidance on parts of explanations which could be skipped by those not interested in mathematical detail, and have tried to give numerical examples of the meaning and use of symbols wherever possible.

VARIATIONS

Hopfield and Hinton, though clearly very influential, are by no means the only contributors to the development of neural computing techniques. In Chapter 9 the well-established work of Teuvo Kohonen of Helsinki University is reported upon, which goes under the heading of 'unsupervised learning'. In this mode, the net discovers hidden patterns in the input data which even the designer of the net may not be aware of. The chapter also considers the applications of such systems to speech recognition and looks at ways in which special neuron models help in the recognition of rotated images.

Continuing with the theme of variations, in Chapter 10 descendants of the WISARD species of neural nets are described. In common with the WISARD these systems are distinguished by being 'weightless' and

dependent on conventional RAM devices. It is shown that what can be achieved with weight-based systems such as Hopfield and Boltzmann models, can also be achieved by RAM-based systems that perform in 'logical' ways. It is shown that this approach forms a basis for the design of highly implementable nets. The object of the chapter is to give the reader the means for designing such nets with specific tasks in mind.

APPLICATIONS

The field of neural nets is very much in its infancy. This means that the numerous researchers who are giving their attention to the new science are applying neural nets to simplified problems to test their ideas rather than solving problems on a realistic scale. In the final sections of this book we therefore concentrate on the principles which will guide the use of neural nets for practical and useful tasks. After all, neural computing will only have 'made it' when computing engineers will begin to use the technique, as one of several tools available to them, in their quest to achieve specific computing tasks. So we try to answer the question, 'what are these tools?'

Because of the closeness of the technique to ideas that come from the structure of the brain, many of the applications that have been studied to date not only attempt to solve engineering problems, but also have an element of trying to explain 'how the brain does it'. It is argued that finding out how the brain does things may lead to the discovery of some efficient engineering principles. After all, the brain does useful things with rather unreliable components and in areas such as speech and vision the brain still seems to set a standard of competence which computers find very hard to equal.

The first application area tackled in Chapter 11 is one of the most successful and proven demonstrations of the ability of neural nets. The task is to take symbolic representation of words as letters and generate phoneme (sound segment) codes that may be used to drive a voice synthesizer. This is the opposite problem to Kohonen's work on speech recognition (Chapter 9) where phonemes were turned into recognized words (Kohonen, 1984, 1988). The chapter continues with a description of the way that a neural net could be made to follow conversations about simple scenes and paraphrase simple stories.

Probably the major application of neural nets lies in the recognition of visual patterns. This is an area where the brain reigns supreme and this chapter is concluded by describing the approaches taken by those who have based the design of vision machines on what is known of vision in the brain. The work of a leading Japanese contributor, Kunihiko Fukushima is reviewed, who has devised and partially built a machine called the 'Neocognitron' which, because it is able to attend selectively