Neural systems engineering: towards a unified design discipline?

The burgeoning interest in 'neural networks' (or 'parallel distributed processing' or 'connectionism') is now an established fact. Whether this will grow into a new engineering discipline or not is still an open question. This article discusses the beginnings of a unified approach aimed at enabling engineers to design and build significant neural systems in an effort to improve the competence of computing systems in general.

by Igor Aleksander, F.Eng.
Imperial College of Science, Technology and Medicine

The appeal of neural networks

The appeal of neural networks stems from the acceptance of the idea that living brains 'compute' in ways which are different from, and are in many ways better than, conventional digital computers. Brains learn to react rapidly to vast amounts of data (from the senses), form inner representations of these perceptions, have a prodigious capacity for 'thinking ahead' and planning very rapidly. Conventional computers do most of these things with considerable difficulty; however they are much better than brains at numerical calculations and at obeying at great speed long lists of rules given by a programmer. A key difference is that, in conventional computing, the programmer must work out in advance *how* the computer will do things while the brain *learns* to perform these computationally sophisticated tasks.

'Neural networks' is primarily an area of study which throws light on *how* the brain may be able to achieve this performance and how it could be reproduced in computing machinery. The methods adopted are rigorous. They include mathematical analyses, computer models and hardware models which use as data the discoveries of experimental neurophysiologists and neuroanatomists. Brain scientists find this exciting as it explains some of their results. Engineers find it exciting as they see ways of improving the competence of computers. It is the latter that concerns us in this article. An outlook (based on the exploitation of the properties of random access memories) which is being developed at Imperial College will be discussed. This is more deeply rooted in digital system engineering than the modelling of brains pursued in many other laboratories.

What is a neuron?

A neural network is composed of a large number of 'nodes', which in the brain (and analogously in artificial systems) are called *neurons*. Each neuron receives information from other neurons through a set of specific connections (called *synapses* in brains) and it 'assesses' incoming patterns of information to decide what new information to transmit through its output (*axon*). Two such neurons are shown in Fig. 1*a*.

In the brain, the transmitted information is coded as a sequence of pulses, the frequency of which appears to carry meaning. In artificial systems a similar coding can be used, although various other ways of representing data are common. For example, it is often assumed that binary signals or encoded numbers are transmitted between artificial neurons. We (Aleksander and Morton,[1] chapters 5 and 10) and, more recently, others [2,3] have taken the view that the neuron is very much like a random access memory. The description relates to Fig. 1*b*. This may be summarised as follows:

The RAM neuron

- The dark circles represent binary 'address' inputs.
- There are, therefore, 2^N possible binary input patterns.
- Each of these patterns addresses a different location in the memory of the RAM.
- Each location stores a B-bit word which is output when addressed; i.e. the RAM can output one of $M=2^B$ messages.
- Each message may be turned into a 'firing probability' of a neuron,[4] i.e. the probability with which the neuron outputs a binary 1.

This is an 'M-probabilistic logic node' (or M-PLN) as described by Catherine Myers,[5] while a similar model, the p-RAM (in which a continuously variable output is assumed to be stored) has been fully analysed by Denise Gorse and John Taylor.[2]

This can be contrasted with classical work in this area which is based on a model of the neuron in which the inputs to a node are 'weighted' and summed to provide the node output. The history of these approaches is summarised below in order to put the points made in this article in context (refer to Aleksander and Morton,[1] chapters 2 and 3, for full details).

Some history

The beginnings

The subject is ancient when measured with yardsticks used in information technology. The neuron was first modelled at MIT in the early 1940s by Warren McCulloch

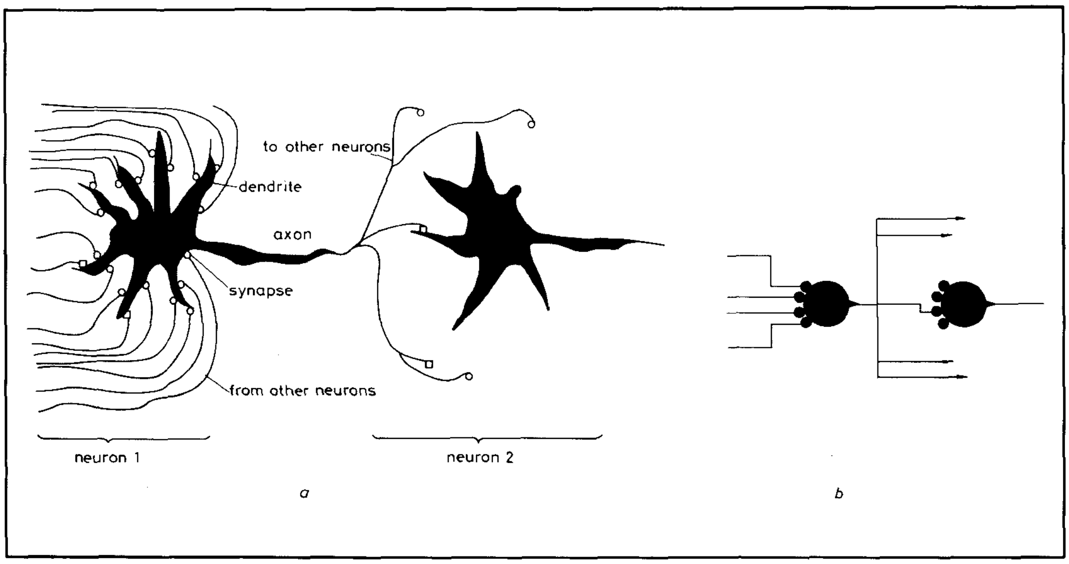

Fig. 1 (a) Two 'real' neurons and their connections, and (b) Schematic representation

(a neurologist) and Walter Pitts (a logician) in terms of a sum of weighted inputs, as mentioned earlier. This predated the seminal definition of the stored-program computer in 1947 by Burks and von Neumann. In the late 1950s Frank Rosenblatt, an electrical engineer at Cornell University, studied neuron models as pattern recognition devices, calling them 'perceptrons'.

The historical flow was interrupted in the late 1960s as a result of the severe criticisms by Seymour Papert and Marvin Minsky of MIT, who, in 1969, showed that, while perceptrons can perform some simple pattern-recognition tasks, they fail to identify some basic properties of such patterns (for example, whether the number of blobs in an image is odd or even, or whether two shapes are connected or not). As such tasks are easily done with simple programs on a computer, many were dissuaded from continuing to study neural nets for the purpose of making improved machines.

The revival

Only a handful of people in the world continued working on neural nets after Minsky and Papert published their book ('Perceptrons: an introduction to computational geometry', MIT Press) in 1969 (see Aleksander and Morton[1] for full references to the work described in this section). In 1982, John Hopfield of the California Institute of Technology published a paper in which he considered an interconnected group of McCulloch and Pitts neurons (with minor modifications) and analysed the ability of such a net to be taught to remain stable for certain patterns.

He drew attention to the fact that such a system had inherent properties (which he referred to as 'emergent'). For example, given a partial presentation of a learnt pattern, the net completes the pattern, thus inferring the whole, though only given some of its parts. It can also resolve ambiguities of the given cue in a statistical manner, and encode patterns in a time sequence. While Hopfield was suggesting that this might lead to silicon chips with useful computational properties, he speculated that these emergent properties may underpin some of the 'intelligence' of higher animals. His paper not only generated a great deal of interest in its own right, but also drew attention to the fact that much interesting work was under way within and outside the USA and that many could add to Hopfield's understanding of neural nets. Indeed, much of this work extended beyond Minsky and Papert's way of defining such nets, making some of their objections less valid.

In the USA researchers calling themselves the Parallel Distributed Processing (PDP) group emerged in 1986 and published a pair of multi-authored books by that name which had an evangelistic effect in encouraging a vast number of research laboratories to initiate work on neural systems. The group focused on the work at Carnegie-Mellon University under Geoffrey Hinton (now at Toronto University) and James McClelland (one co-editor, the other being David Rumelhart of the University of California at San Diego) with contributions from Terry Sejnowski of Johns Hopkins University and several others in the San Diego area. Other well-known scientists (including Francis Crick and Donald Norman) contributed to these books, commenting on the importance of the emerging field. There were frequent references to the work of Steve Grossberg at Boston who had been active in the field for over a decade and Bernard Widrow, a leader in this field in the 1960s and still going strong.

Outside the USA, it became recognised that the work of Teuvo Kohonen (Finland), Edouardo Caianiello (Italy), Rolf Eckmiller (Germany), Chris von der Malsburg (Germany), Igor Aleksander (UK), John Taylor (UK), Shun-Ichi Amari (Japan), Kunihiko Fukushima (Japan) and many others was concerned with many of the issues raised by the PDP group and had, in some cases, been present in the literature for some time. The revival is not in doubt, but what is its engineering significance?

Circuit paradigms

A way of summarising these developments in an engineering sense is to look at the major structures that have emerged from the above research effort. These are shown in Fig. 2.

The single-layer perceptron

The 'perceptron' structure is shown in Fig. 2*a*. Essentially, it is composed of several 'large' neurons which perform a pattern-recognition task.

Each neuron is trained on a different class of patterns, and, when an unknown pattern is presented to the system, the neuron which responds most strongly determines the classification of the unknown pattern. So, if the task is to recognise the hand-printed alphabet, 26 such neurons would be required.

The multi-layer perceptron

Shown in Fig. 2*b*, the multi-layer perceptron is the workhorse of many current neural net applications. It overcomes the objections discovered by Minsky and Papert through the introduction of 'hidden units' between the input and the output units. The units in this hidden layer act in an auxiliary fashion, and it can be shown that having enough of them solves any problem. This hidden layer has to be trained through a process of *error back propagation*, which is somewhat cumbersome and time-consuming. An engineering disadvantage is that it is also not always clear as to how many units should be used in the hidden layer.

The Hopfield net

The major characteristic of the Hopfield net (Fig. 2*c*) is the fact that every node is connected to all other nodes (excluding itself in Hopfield's formulation). The system operates by storing certain trained patterns (states) to be stable in the feedback paths (state variables). The key emergent property of the scheme is that it completes partial presentations of stable patterns, provided that the presentation is not ambiguous. It does this in fewer steps than one could execute a template-matching search in conventional programming. This enables the system to be 'associative', both in the sense that incomplete patterns may be completed (e.g. unmasking a criminal who was covering part of his face when 'caught' by the video camera during a bank raid), and in the sense of associating the image of a piece of Brie with its attributes (cheese, runny, strong smell...).

The Kohonen net

The dark line shown in Fig. 2*d* draws attention to the central feature of many of Kohonen's approaches and that is to introduce a spatial meaning into the net. The Figure shows a two-dimensional connection (two elements joined by a line are 'neighbours'). Obviously, other neighbourhood structures can be imagined. The net (often in an 'unsupervised' way) is made to fire locally for a given class of inputs (such as a phoneme in speech recognition). A particular sequence of such recognitions (e.g. a spoken word) becomes associated with a characteristic firing 'trajectory' which exists within the geometrical space of the net.

There are many other circuit and system structures which have interesting engineering properties. Grossberg's Adaptive Resonance Theory (ART) networks[6] and Fukushima's Noecognitron (see Reference 1 for details) should be of particular interest to engineers, as they have highly characteristic structures which are beyond the scope of this article.

Current concerns

Before going on to argue that a unified engineering approach to neural systems engineering is necessary, and how it might be achieved, it is pertinent to look at the current concerns of the research community. This is a highly eclectic group. Apart from neurophysiologists, computer scientists and engineers, it attracts mathematicians, physicists, psychologists and, on the applied side, financial forecasters, designers of quality monitoring equipment in manufacture, and members of the communications and defence industries.

There appear to be three major pursuits which occupy the community. There are those who wish to *apply and implement* neural net techniques, those who wish to develop the *theory* further and those who are developing *biological models* which seek to explain the findings of experimenters in the life sciences. Each of these activities has further ramifications.

Applications

Some sectors of the computing industry have reacted rapidly in producing software that enables a user to set up a neural net (usually a multi-layer perceptron), train it and get a 'feel' for what it does and how changing its structure leads to changes in performance. Such systems are usually limited in size, and support nets of only 100 or so nodes. To set this in perspective, it is felt that neural networks will only become widely used when technologies that provide hundreds of thousands of nodes become available. Nevertheless, small nets are being applied to problems in which not a great deal of data has to be recognised. Financial forecasting is a favourite example where, say, the price of assets is forecast as a function of half a dozen or so economic variables. The net is trained with data for some period in time and tested on other data from some other period in time to assign a level of confidence in its performance.

Most research in applications is done by direct simulation on

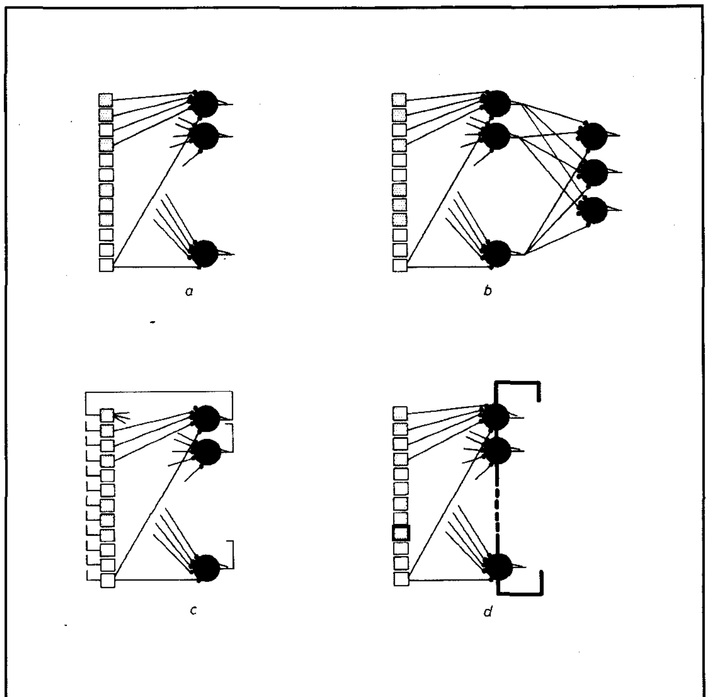

Fig. 2 Four circuit paradigms: (*a*) Single-layer perceptron; (*b*) Multi-layer perceptron; (*c*) Hopfield net; and (*d*) Kohonen net

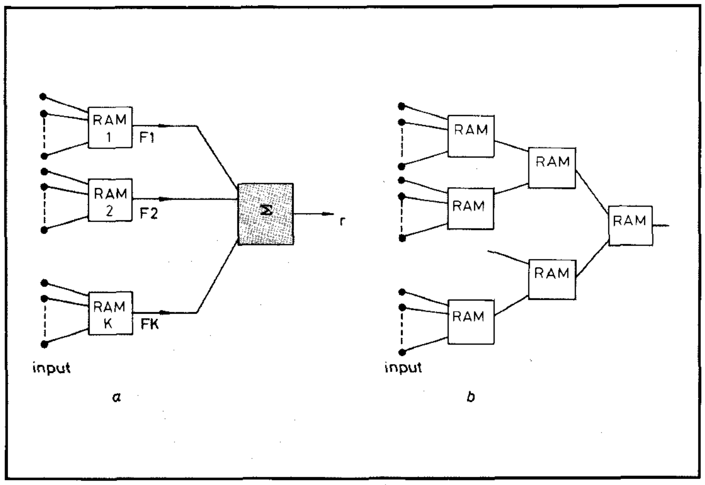

Fig. 3 Large neural nodes: (*a*) A discriminator; (*b*) A pyramid

conventional computers. Speech processing, natural language understanding and vision are favourite areas of attack. Researchers are not so much trying to develop working systems as aiming to understand the effectiveness of various structures and training strategies. There is much to be learned: it is clear that a single net is not going to achieve much competence in these areas. The major target for researchers is to represent, in layered systems of networks, the knowledge that is the basis of human cognitive competence. This may not only produce computing machines that have a friendly, human-like interface which uses natural language, but may throw some light on the way that such tasks are being achieved by the neural structures, the brain. An example of such 'reverse engineering' of human cognition is given later in this article.

Implementation

The future of neural nets does not lie in simulations on conventional machines. Equally, it is not likely that specially designed neural hardware will entirely replace computers as we know them. Engineers and technologists are mainly interested in a good mix of conventional and neural technology. Currently many companies are manufacturing fast neural 'cards' that can be plugged into conventional machines. This allows owners of even quite modest PCs to experiment with small nets of a few hundred neurons.

Some research laboratories have used parallel systems of many processors to investigate systems of thousands of neurons, while others have managed to study nets with hundreds of thousands of neurons. The WISARD system, first developed in 1981 by teams now at Imperial College and Brunel University and engineered in industry in 1984, is an early example of a system that contains 250 000 simple neurons (see Aleksander and Morton,[1] chapter 5). Silicon-chip manufacturers too have become deeply interested in turning neuron models into very large scale integrated (VLSI) systems. There is much debate on how best to do this: should systems be analogue or digital? Should they be like living neurons or should they be optimised for what can be done in silicon? This is an interesting race to watch.

Theory

The training of a neural net, particularly one that has internal or 'hidden' neurons is still a lengthy and haphazard business. The hidden neurons are required to develop auxiliary functions which allow the other or 'visible' neurons to perform desired tasks. As mentioned earlier the technique that is called error back-propagation is dominant. It centres on transmitting error messages backwards through the network and adjusting the functions of the hidden neurons so as to reduce such errors. The method is clumsy and can get stuck in some seemingly minimum error only to discover that there is an even lower error if one were to start the process again. Many are working on improving on this technique and developing alternative methods.

Other theoretical concerns include investigations of the way special structures, particularly layered ones, operate. As has been said, if neural nets are to achieve competence as 'knowledge stores' they will not be randomly connected. The corollary to this is knowing how to put together sub-nets into large nets in order to achieve specific learning behaviours. This is an area where theory mixes well with engineering design and much imagination. This is discussed at greater length later in this article.

Biological models

Having learnt to analyse and design neural systems, it becomes attractive to apply the same methods to the analysis of living systems. The human may be a little too complex to attack at present, so researchers have looked down the complexity scale. As an example (among the very many others: too many to cite in this article), Catherine Myers[5] at Imperial College has settled on the octopus as a suitable case for modelling. The octopus has a sufficiently interesting range of learning behaviours, and a sufficiently well-studied neural structure to try to put the two together by reverse-engineering its brain. Such work leads not only to remarkably good matches between the behaviour of the live and that of the artificial, but also to sets of important questions from which neurobiologists can develop further experiments.

At the more detailed level, the neuron itself comes under close scrutiny. Living neurons are likely to have much more complex characteristics than those suggested by McCulloch and Pitts which are widely used at present. John Taylor of King's College in London and Denise Gorse of University College in London have been successful in showing that systems of probabilistic versions of random-access memories (p-RAM) behave in a way that closely resembles their living counterparts. In fact, this is a 'weightless' model of a neuron, in which the function is altered by changing the content of an addressable memory rather than the adjustment of connection strengths. Similarly weightless models were earlier developed (largely by the Imperial and Brunel groups) to improve neural machine design, and it may be no coincidence that what the technologist finds useful for artificial systems has also been found to be an improved biological model. John Taylor has gone on to show that mathematical analysis, deeper than that used by the PDP group, can provide a clear explanation of the functioning of intricate parts of living visual systems.

RAM-based circuit paradigms

Here the notion is introduced that, through the RAM paradigm, a general component can be developed, the general neural unit (GNU), on which the engineering of significant neural systems can be based. This needs to be approached in two steps: first an assessment has to be made of cost as a function of number of inputs for large neurons, and then a specification of the GNU is created which aims to encompass the properties of a large diversity of systems (as described in the section on circuit paradigms). The selection of specific properties then is affected by the selection of the appropriate parameters of the GNU.

Large neurons

An exact comparison between the 'weighted' McCulloch and Pitts node and the 'weightless' RAM nodes discussed above reveals the following advantages of the latter:

- Silicon technology for making RAMs already exists, whereas the technology for making analogue weights still needs much research. Digital methods for making weights are clumsy and wasteful of memory.[4]
- As shown by Gorse and Taylor,[2] the RAM model is capable of modelling sophisticated behaviours that are known to exist in living neurons (e.g. noisy responses and multiplicative effects between synapses) which cannot be captured by the conventional, 'weighted' method.

As against this, seeing RAMs as single neurons has several disadvantages:

- The memory cost of the RAM scales up as 2^N with the number of its inputs N, while weighted nodes scale up as N^2 assuming digital weights.
- The RAM itself only responds to data on which it was trained; i.e. it does not generalise this information to similar data (although there are ways of getting around this by using noisy training patterns or allowing data to spread to other RAM locations—as in a device called the G-RAM[7] (generalising RAM)).

It is of some interest that both these disadvantages may be overcome by the same piece of circuit design: the use of several RAMs to achieve the function of one neural node. These are shown in Fig. 3.

Fig. 3*a* shows a 'discriminator' as used in the WISARD system. In this system a reconfigurable architecture is used and the size of the node is determined by the user. The entire system is bounded by a fixed amount of memory and known applications of the system have used two nodes with 256 000 inputs each broken up into 32 000 8-input RAMs (face expression recognition) and, in another case, 1024 1024-input nodes of four-input RAMs (for image-to-image transformations). A discriminator has a well-known generalisation characteristic which decreases exponentially with the size of the RAMs. The memory cost scales up $(N/n)2^n$, where n is the number of individual inputs to each RAM.

Fig. 3*b* shows an alternative way of reducing storage: the pyramid of RAMs. The difference between this and the discriminator is that the pyramid can perform accurate local operations (e.g. discover a missing bit in the area sensed by a RAM in the input layer) whereas the discriminator is sensitive to overall similarity and will miss sharp detail. Again, details of these two methodologies have been published in Aleksander and Morton,[1] chapters 5 (discriminators) and 10 (pyramids).

The conclusion is, however, that large neurons can be made with existing VLSI technology. Whether in an engineered system both pyramids and discriminators are made available remains an open question, the answer to which lies in applications research which is currently being pursued. Given that this question will be resolved, it is proposed here that a general unit be defined which contains these large neurons but leaves the designer to think in broader system-structure terms as will be described.

The General Neural Unit (GNU)

The GNU is shown in Fig. 4. The units (such as the one marked A) are large weightless nodes designed through one of the two methodologies discussed in the previous subsection. The designer need merely bear in mind that these nodes will behave as neurons with a specified degree of generalisation. The GNU has *functional* parameters, of which two of the major ones are summarised below.

Association: It is assumed that a unit receives inputs from a proportion Q of other units. So, a selection of Q from 0 to 100% takes the system from having the properties of a single-layer perceptron to those of an autoassociator. The exact effect of intermediate values of Q is the subject of current study. The properties of the net are also sensitive to whether the feedback allows self-connection in the nodes or not. This too will be discussed in more detail in future publications.

Temporal properties: The feedback between the nodes gives the unit temporal properties. There are two kinds: *input* and *output*. Input temporal properties refer to the fact that the net can be sensitive to the order in which input messages are applied. Output temporal properties refer to the possibility of the net stepping through a sequence of states while the input is held constant. Clearly, which of these properties is in force depends on the way that the net is trained.

To illustrate the framework in which a neural designer might operate, a design example is discussed next.

Design example

This design example is included for the purpose of illustrating a methodological framework only: it is completely fictional in all other respects. The target is to develop a system that investigates a visual buffer containing circles, squares and triangles, finding one such object by describing its position with a voice synthesiser or describing the scene in terms of the position of all the objects. So the input is constrained to the following two sentence types:

- FIND <CIRCLE/SQUARE/TRIANGLE>
- DESCRIBE

Fig. 5 indicates an attempt at a solution. The emphasised oblongs are the GNUs. The process of reasoning that leads to this scheme will now be described.

General operation

The user issues a command (either by speaking, in which case a speech-to-phoneme net is required, or by typing the words on a keyboard). This is detected by the word recognition (WR) GNU which has a unique state

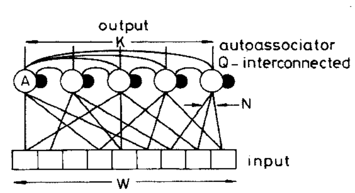

Fig. 4 Generalised neural unit

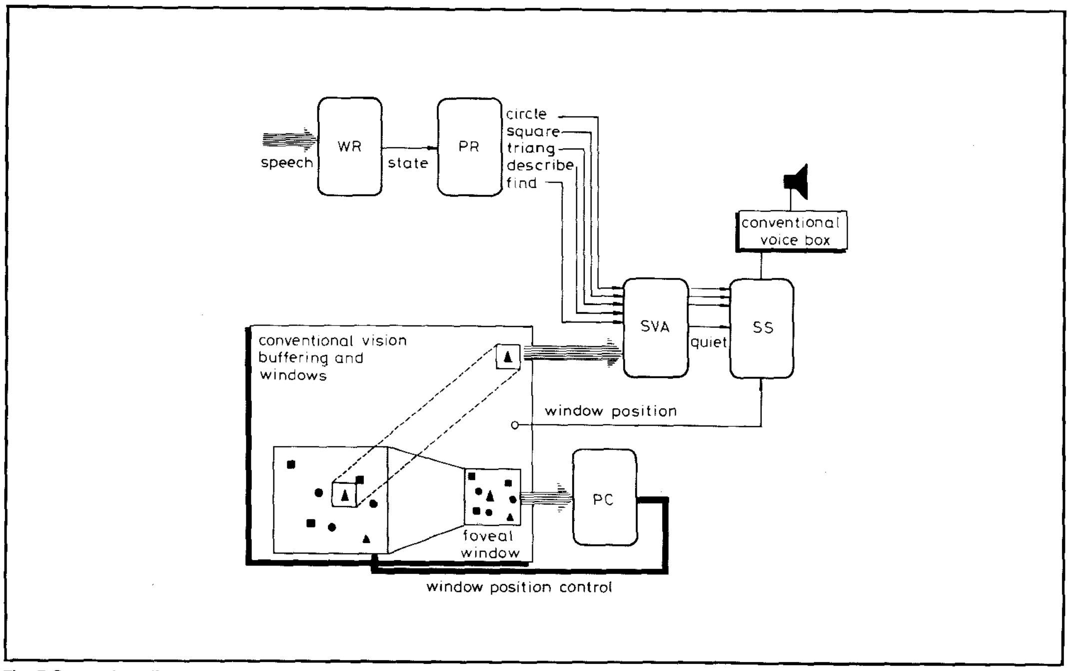

Fig. 5 Scene describer

for each word. The grammar of the input words is very simple:

s: = Find *x*/Describe
x: = Circle/Square/Triangle

s is the starting symbol, *x* is the only other non-terminal symbol, and the rest of the words are all terminal symbols. So there are only four possible sentences in the grammar.

The phrase recognition (PR) GNU translates the state of the WR into firing patterns on five terminals, one for each word. Note that a phrase such as 'Find Circle' should cause firing at two of the PR outputs. This is passed on to the speech/vision association (SVA) GNU, which also receives an input from a window generated by a buffer that holds an image of the scene. The buffering arrangements are executed by conventional means. However, a foveal window is made to move from object to object through the action of the position control (PC) GNU which is trained to output the window position increment co-ordinates to move to another object in the foveal input to the PC itself. Note that the window which provides input to the SVA is not foveal, but presents single objects in its field.

The SVA, by association, finds the code that is appropriate to the window-centred image and passes this on to speech sequencing (SS) GNU. This also receives position signals from the vision buffer, and outputs a sequence of signals received by a conventional 'stored speech' box which might be preprogrammed to output synthesised speech such as 'Circle at *X* 5 and *Y* 10'.

During a Find *x* operation the SVA is trained to operate slightly differently, in the sense that it does not issue an output until the image matches *x*.

This very rough description leads us to appreciate some of the parameters of the GNUs.

GNU characteristics

WR—Word Recognition: Input temporal sensitivity is required, as a sequence has to be transformed into a state. This means that a non-zero *Q* is required.

PR—Phrase Recognition: Short input temporal sensitivity is required so as to be able to pair words such as 'Find Circle'. Hence, non-zero *Q* is required. Also the length of time that the output persists controls the point at which the whole operation stops.

PC—Position Controller: This requires no temporal sensitivity. However, it needs to be probabilistic in the sense that the choice of which should be the next centred object should be arbitrary. This would prevent the system from getting into a two-state cycle between two objects and would ensure that most objects would be visited (perhaps more than once) during a Describe operation. A zero *Q* could be used in this case.

SVA—Speech/Vision Associator: There are no temporal requirements, but a non-zero *Q* is required to perform the necessary associations.

SS—Speech Sequencer: This GNU requires output temporal capacity, in the sense that it needs to produce a sequence of signals to drive the voice box. Again non-zero *Q* is required.

Comment

It is recalled that this example is there solely for purposes of illustration. It is doubtful that a system of this kind would work without a great deal more thought being given to the details of its training and structure. But the main illustrative point is clear. The designer can express himself in terms of GNU parameters and it is a mastery of knowing the effect of changes in these that underlies much of the future art of neural net design, and indicates the direction of current research.

Future applications?

Given the successful development of engineering design skills as advocated in this article, what applications can be enabled that cannot be achieved now with conventional methods? Books could be written on this topic, only a telegraphic statement will be made here, separated into timed predictions.

Cognitive computing (years 7-10)

A general-purpose machine that has hitherto unattained abilities in understanding natural language, understanding scenes, and is directed towards office automation and management applications.

Life-preservation (years 7-10)
Similar to cognitive computing but with well defined dedicated systems: e.g. automatic air traffic control; medical screening systems; policing systems; vehicle safety systems.

Consumer products (years 5-7)
Product development of vision-dependent devices, e.g. content-addressed interactive video. Speech driven devices: e.g. washing machines, video programmers. Speech generation devices: e.g. alarm and surveillance systems, performance monitoring in cars.

Manufacturing products (years 5-7)
Similar to consumer products, but directed towards production equipment. Hands-free operations, mainly, automatic inspection and quality control.

Finance (years 2-4)
Development of software products that run reliably on conventional machines: forecasting, acquisitions, risk assessment...

Interdisciplinary education?

Educational history is strewn with attempts to foster interdisciplinary topics. Typical and reasonably successful have been subjects such as cybernetics and system science. These have in common a desire to analyse a large number of mechanisms in the world using a unified selection of theoretical tools. The interdisciplinary appeal of neural networks is quite different. As a subject for study it exposes the student to aspects of the expertise of life scientists, engineers, computer experts, natural scientists, mathematicians and psychologists in order to understand a specific set of phenomena: the emergent properties of brain-like systems.

Despite its long history, the state of this topic is at an early stage. But the excitement it now generates has the character of the beginnings of a new and important academic subject with strong engineering aspects. Certainly the subject has a great appeal among existing students. The opportunity should not be missed to add the topic to the curricula of the supporting disciplines. It could have a significant effect in attracting the bright and the young back into engineering studies.

References

1 ALEKSANDER, I., and MORTON, H. B.: 'An introduction to neural computing' (Chapman and Hall, 1990)
2 GORSE, D., and TAYLOR, J. G.: 'An analysis of noisy RAM and neural nets', *Physica D.* 1989, **34**, pp.90-114
3 JUDD, J. S.: 'Neural network design and the complexity of learning' (MIT Press, 1990)
4 ALEKSANDER, I.: 'Are special chips necessary for neural computing?' *in* DELGADO-FRIAS and MOORE (Eds.): 'VLSI for artificial intelligence', (Kluwer, 1989)
5 MYERS, C.: 'Learning with delayed reinforcement in an exploratory, probabilistic logic neural network', PhD thesis, University of London (Imperial College) 1990
6 GROSSBERG, S.: 'Competitive learning: from interactive activation to adaptive resonance', *Cognitive Science*, 1987, **11**, pp.23-63
7 ALEKSANDER, I.: 'Ideal neurons for neural computers', Proceedings ICNC (Springer Verlag, Dusseldorf, 1990)

Prof. Igor Aleksander is Head of Department and Dennis Gabor Chair of Electrical Engineering, Department of Electrical Engineering, Imperial College of Science, Technology & Medicine, Exhibition Road, London SW7 2BT, UK. He is an IEE Fellow