Analog Neural Networks For Real-Time Systems

By Gary Dan

December 14, 1992

CS 621

Preface

It is interesting that people generally marvel at the latest computer gadgetry and can't wait to play with it. Yet most of us possess devices that are far more complex and far more powerful than any of these gadgets and we rarely *play* with or try to understand it. A popular comedian, Emo Philips, once said, "I used to say that my favorite part of my body was my brain, until I realized which part was telling me that".

I have always been amazed at what our brain does. Look out your window and see how fast your brain is able to recognize specific objects, even if only a fraction of it is visible. Now consider that while you were looking out the window, your brain was also maintaining body functions such as balance, eye control, respiration, a host of other processes while you were probably concentrating on something unrelated. We are very good at working with objects but do poorly at raw numerical calculation, of course our brains developed machines to make up for that deficiency.

What is thinking? Philosophers, Psychologist, Neural Biologist and Computer Scientist have tried to answer this question to no avail. Do you *think* of thinking as a chemical processes in the neuron or take a macro approach of clusters of neurons working together. Can we figure anything out by thinking of how we think? Some may say that religion belongs in this discussion. One must remember that religion is a *result* of thought. These questions must be better understood if we are ever to make autonomous machines.

Curiosity about how our brain works and a fascinating article in Byte magazine [2] about Analog Neural Networks were the driving forces that got me interested in studying neural networks and of course, writing this paper. This is a very interesting subject and I plan further investigations. Five weeks ago I hadn't the vaguest idea what a neural network was, let alone an analog neural network, now I know enough to get me by at a cocktail party. I enjoyed writing this paper!

Abstract

The animal brain and in particular the human brain is a fascinating piece of computer architecture. Researchers have tried to mimic the functioning of the "biological computer" and science fiction has long fantasized machines with capabilities similar to humans. We are still nowhere near the wishes of those grand plans but we are beginning to understand some of the functioning of the brain enough to loosely design analog neural networks with some of the massively parallel and non-linear characteristics. There are many types of neural networks; software implementations of neural networks, artificial neural networks, which are digital architectures (ie digital processing elements) and analog neural networks. We will look at the analog neural networks in detail along with an overview of the previous two for comparison. We will then look into Real-time applications and performance of the analog system, and compare when possible performances of the digital and software implementations. Not all processes that lend themselves to neural networks need to be analog, in fact many do not and many can not. Upon surveying these systems, the real time system designer should be able to make an informed decision as to whether an analog system would be appropriate and perhaps which system is most suitable for a particular application.

Introduction

We sometimes take for granted the enormous processing power that the human brain possesses. Sure, we are not particularly good at number crunching but with object manipulation we know of no equivalent. The neurons in the human cerebral cortex fire at a rate of 1,000 times per second [3] giving a 1ms time constant; as an individual component, this is very slow. The Honey Bee [2] which has about $1/100{,}000^{th}$ of the 100 billion neurons in the human brain is on the order of 1,000 times faster (10,000 GFLOPS [2]) than todays most powerful computers for the tasks it performs. Compare the size of the Honey Bee's brain to a super computer that is kept in a climate controlled room and imagine the differential in power requirements. Yet the Honey Bee's brain controls flight, balance, navigation and behavior through sophisticated input devices. How is this done in nature? Through massive parallelism each neuron, slow as it may be, works in parallel with thousands of other neurons. Although digital computers have various degrees of "parallel" architectures, for example the Connection machine may be considered highly parallel, they don't come close to the parallelism achieved in biological networks and analog neural networks.

The goal in developing high performance Neural Networks is to perform real time analysis on 'fuzzy' data where the digital preciseness is not a requirement and the information provided to the system may not be well known or nonlinear. Such system are pattern and speech recognition, avionics and control systems.

What Is A Neural Network

A neural network is a set of interconnected nodes (neurons and synapses) whose state is on or off (or can even have in between states) depending on the result of summing all inputs.

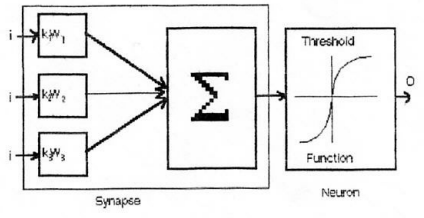

Figure 1, Model of neuron.

Nodes have one output that can theoretically connect to every other node in the system and therefore a node can have many inputs. Inputs to nodes are weighted in response to a learning mechanism. Determination as to whether a node fires is a function of the summation of the weighted inputs applied to a threshold function $y = f(Sk_iw_i)$ where, S is the summation function, w_i is an input, k_i is a weight and f() is a threshold function, see figure 1.

The threshold function is generally a sigmoidal function, $f(x) = (1-e^{-x})/(1+e^{x})$, although a piecewise linear function can be used. The sigmoid is generally preferred because it better resembles the biological neural net.

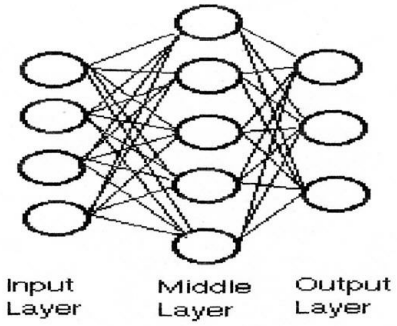

Figure 2, Three layer network.

The most difficult part of understanding a neural network is abandoning the notion of a conventional computer. A digital computer operates sequentially on instructions while a neural network receives input, reacts and learns. Output from a neural network is a series of on and off states from the output layer of the network. How these results are interpreted is good topic of study. In general a digital computer, hardware (such as a control system) or even another neural network can handle the output.

Nature's Neural Network

The human brain with it's 10^{11} neurons has on average 1,000 synapses for each neuron giving 10^{14} connections [4] although in [14] a single neuron may connect to as many as 10,000 other neurons. Such large numbers of connections gives the biological network a high degree of fault tolerance. A neuron's activation depends on the input of many other neurons, therefore if a few go bad, the system will not be degraded [4]. The biological neuron is composed of the dendrites, axon, nerve body and a synapse to connect dendrites to axons. It is beyond the scope of this paper to delve into the details of the physiology of the biological neuron other then to parallel its analog equivalent.

The dendrites carry signals into the nerve body and the axons carry signals away. The synapse, as mentioned above, conducts between axons and dendrites and provides for the temporal nature of signal transmission. The ramification of the time delayed response is not clearly understood at this time with respect to learning. The dendrites act as a signal enhancer xor as an inhibitor. Each signal is weighted in the synapse and if the sum of these weighted signals reaches a threshold value the neuron fires, otherwise nothing. If the neuron fires, then a pulsing signal is sent through the axon. Changing the weighted factors and the thresholds constitutes learning; how this is done biol2ogically is another matter. The synapse and dendrites perform the weighting and summing of the inputs and apply the non-linear threshold function to all of the inputs. Frequency and not the magnitude of the neuron output determines the importance of the signal.

The highly interconnected nature of the biological neural net, and that neurons don't store information other than weighted values, leads us to believe that information (learning) is stored throughout the network (brain). That is, it is distributed collectively over large parts or entire network, depending on how big the network is. From the biological standpoint we see revealing instances when a person suffers a head trauma. Neural networks are inherently prone to small degrees of error; individual component failures or errors will not adversely effect the system since results are based on groups of neurons and not one single neuron. In other words, neural networks are highly fault tolerant. But when large numbers of neurons are effected, many functions of the brain may also be effected. To emphasize the point that information is believed to be stored collectively throughout the network, there are cases (two specifically), where the inner $1/3^{rd}$ to 1/2 of a persons brain was lost due to post trauma swelling resulting from a head injury. To the casual observer, it was barely noticeable that such traumatic injury had taken place. That certainly does not mean that there was not profound damage, but

much of the network did survive. On the contrary most post trauma head injuries do suffer tremendous damage and a lot of collective information is lost. Relearning is possible but rarely complete due to the shear number of neurons lost.

Analog Systems

The analog system is directly modeled from the physiology of the human cerebral cortex, generally considered the center of thought. Although it is not thoroughly understood how animals learn, it is hoped that by mimicking the animal brain and with further advances in neuroscience we can begin to apply these processes to machines. Like it's biological counterpart, accuracy is not the forte of the analog neural network. If numerical precision is required, then the analog system should not be used.

What is believed to be an analog equivalent of the biological neural networks has been built with VLSI technology. Of course the biological neuron is not fully understood and is far more complex than the analog equivalent. Previous attempts were made in the 50's and 60's but without VLSI, the fabrication of practical systems was next to impossible. The components of the analog neural network, like it's biological model, include the neuron, synapse and switching circuit. A time constant module is also included to simulate the temporal nature of the biological neuron. Neuron modules are laid out next to synapse modules and switches modules are used to route the signals to the synapse modules. A host digital computer is used to monitor the network and to program it. Any neuron, synapse, switch or time constant module can be accessed by the host computer using multiplexers. Programming involves setting synaptic gains, adjusting the threshold function and setting the connections. For feedforward type systems, a hands on approach to learning (programming) is necessary as opposed to backpropagation and other self adjusting algorithms. The interaction of the host computer is completely non-evasive. That is the neural net is unaware of its existence except for uninterrupted changes in weighted constants and the threshold bias. There are some highly experimental systems that incorporate self learning [6] but most systems have a learning mode and an executing mode.

In the general case of analog neural networks, every neuron can connect to every other neuron and all weights and threshold functions can be adjusted. With current technology a scaled version of the general case must be implemented because of fabrication problems in silicon chips. In the human brain, one neuron typically connects to as many as 1,000 to 10,000 other neurons [14]. For the system built in [13] each neuron can accept up to 32 signals and learning algorithms are implement by the host computer sampling neuron output and resetting weighted synaptic values.

The synaptic weights are set by a discrete digital signal from the host with a typical resolution of 3-5 bits. In [13] they range in value from 0-10 with a resolution of 5-b with one bit for the sign. Synaptic time constants are set by a 4-b control word. The time constant, if set, maps to a range between 5 and 1,000ms and can be used to reduce cross talk.

The neuron, unlike it's biological counterpart, sums the weighted synaptic signals and applies a threshold function (typically a sigmoidal function because of the asymptotic limits and the nonlinearity of the function better assimilates its biological counterpart). Summing and weighting in the biological network is performed in the synapse. If the weighted sum exceeds the threshold, then the neuron fires *continuously*. Unlike it's biological counterpart, the importances of a signal is the signal strength and not the pulsed frequency. Learning involves changing the synaptic weight, for when you change the weight, the system responds differently to the same input.

The switching circuit changes the topology by routing a neurons output to any other neuron in the system. In [13] each neuron can connect to as many as 32 other neurons. Switches are set by the host digital computer and should not be changed once the network is operational.

An Analog Architecture

The architecture of [4] will be used as an example of an analog neural network. A second analog architecture with self modifying synapses will be discussed afterwards. A modular architecture with a programmable topology, as shown in figure 3, contains arrays of neurons, synapses, time constants and switching circuits and host digital computer to adjust the characteristics of these modules. The neuron module contains eight neurons and a multiplexer so the digital host can sample any neuron in the module without interfering with the network.

Neurons have a current input and a voltage output since currents can be easily scaled and summed while voltages can fan out. If the input current to the neuron exceeds a threshold value then the neuron outputs a voltage between 0 and 4V. Magnitude and not pulse frequency determines the relative importance of the signal. The synapse module is designed to have 16 inputs and 8 outputs per module; it has a voltage input and a current output. There are two synapse modules connected to

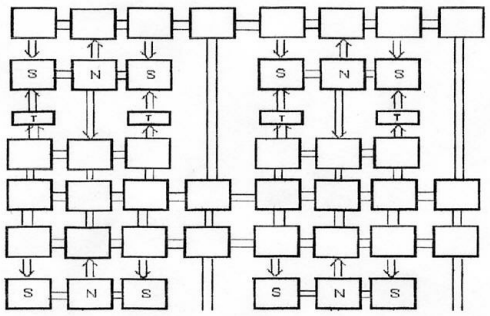

Figure 3, Block diagram of a general neural network [4]. N - neurons, S - synapses and T - Time constants. Blank boxes are switching circuits, [4].

each neuron module for 32 inputs to the neuron. The inputs are converted to a current, scaled, summed and sent to a neuron. Gains for the scaling circuitry are set by the host digital computer and can be changed in real time. The time constant modules can be set by the host computer for values ranging from 5 to 1,000ms. Applications such as acoustics and pattern recognition usually need to be slowed so that the system can settle and output can be sampled before the input pattern changes. In essence, the time constant sets a sampling rate. The final module to discuss is the switch. Switches are set by the host and connect horizontal lines to vertical lines, see figure 3. All of the modules have some sort of local memory (a few bits) to store settings from the digital host.

The second analog architecture to be discussed is based on self learning circuits. In the previous discussion, learning was accomplished by the host digital computer sampling the output and, hopefully, adjusting the weights appropriately. In the architecture described in [6], circuits are built such that they autonomously change the weights depending on the magnitudes and rates of the stimulus. Such forms of non-associative learning are called sensitization. In this system, when a high magnitude stimulus is given frequently, special synapse circuits retain capacitance and modify the weights of the whole synapse. These systems are currently hardwired for specific responses, as is nature (consider the pain reflex). They are still highly experimental and not ready for real world applications.

The analog neural network does not need a clock to synchronize events; processing occurs continuously and in parallel. Time is not a discrete event as in the digital computer, rather a continuum. Generally, the larger the network the faster the steady state response at the output nodes, less propagation delays.

Digital Systems

When we talk about digital systems for neural networks, we mean parallel architectures with processing elements that attempt to digitally simulate the biological neural network. Compelling reasons to use digital architectures are: if a high degree of accuracy is required and digital logic is established and readily available. There are

numerous architectures for the digital neural network. The architecture shown in figure 4, consists of a RISC processing unit, PU, a processing element, PE and local memory. Together, each processor represents a node. The PU's operate in parallel on sequential instructions. They send and receive information to other nodes through the PE's. The PE's in most systems is shared (global) memory. Some local memory is available for calculations and the state of the PU. The digital system computes thresholds and sums inputs digitally. Also, the node output is a digital word. These systems are not, proportionally, as highly connected as their analog counterparts. For example, in [3], the Anza Plus has 1,000 PU's, 1,000 connections and a processing speed of *6,000* connection updates per second. PU's can be implemented using signal processing chips or industry standard microprocessors. Digital and analog circuits can be combined, as in [8]. That system can perform *3.33 billion* connection updates per second. This is a good resume for the analog system.

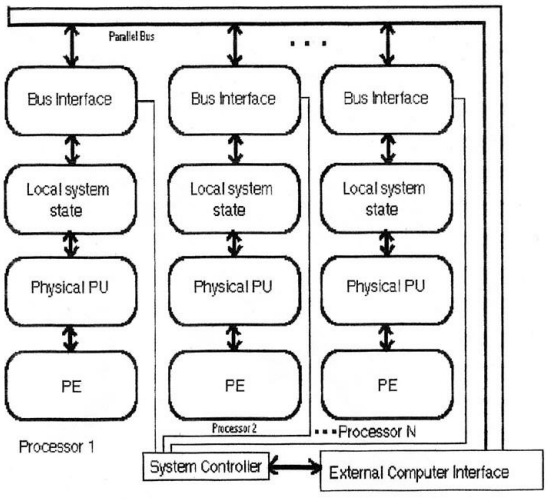

Figure 4, General digital architecture, [3]

Software Implementations

Writing software to simulate an analog network is currently the most prevalent method of implementing neural networks. Hardware for the analog and digital systems is very expensive and many applications do not need the speed that the latter two offer. Using any programming language that allows recursion and preferably dynamic memory allocation, one can simulate a neural network. The drawbacks are numerous. Calculating the threshold function is time consuming and sending a nodes output to every connected node has a tremendous amount of overhead. The nodes are accessed one at a time and in order for a node in a successive layer to activate, all previous connecting nodes must have been accessed. None the less, there are some real-time applications using software implementations.

Learning Algorithms

The majority of the neural networks have a learning phase, off-line, and an on-line phase. There are some hardware neural networks that implement learning mechanisms that attempt to copy the biological neural network and learn in real-time while on-line. They are mostly experimental and not widely used at this time. Response times and learning rates will vary with initial inputs and initial weights.

When a system learns, it is given a set of inputs and a correct set of desired outputs. By manipulating the weights, the system can force it's outputs to match the desired output. It is important that the input set is eclectic, otherwise the network could eliminate previous learned responses. For example, if one is teaching a network to recognize handwritten digits and you teach it all the variations of the digit '2', when you proceed to the next digit, the weights set for the digit '2' might be changed so much that the network *forgets* the digit '2'. The best method is to take a random sampling of all of the digits and variations of the digits, cyclically.

A single neuron neural network with two inputs is desired so that, if the sum of the inputs is between 0.4V and 0.5V then the neuron will activate, send a signal. The weights are initially set arbitrarily. The first set of inputs arrives and the sum is 0.6V, yet the neuron is on. The trainer sets the weights so that the neuron is off. Now a sum

of 0.45V arrives and the neuron is still off. The trainer again adjusts the weights until the neuron is on. A third input arrives with a sum of 0.51V and the neuron is on, so the trainer again adjusts the weights. This tedious process continues until the network exhibits the proper response for all inputs. For this particular example, it is likely that more than one neuron will be needed.

A synaptic topology, *architecture*, that implements a cyclic directed graph can implement a gradient descent algorithm autonomously. A gradient descent architecture allows neurons to pass error information back to previous neurons. This method is not employed in the biological network, rather an unsupervised trial and error method is used, neurons cannot pass information back. Acyclic synaptic topologies implement feedforward algorithms and synaptic weights must be adjusted by a supervisor (trainer). Unfortunately for the feedforward architectures, there is no well defined algorithm that states how to adjust the weights of the inner, *hidden*, layers and therefore the number of hidden layers must be kept small (one or two). If there were, for example, five inner layers, it would be very difficult to determine which nodes were causing errors to propagate. Adjusting the inner nodes would be a hit or miss situation.

The most prevalent learning mechanism for architectures that support gradient descent, is backpropagation. Backpropagation has two phases; the forward phase and the backward phase whereby the weights of the previous layers are adjusted based on the output layer's error [15]. Problems associated with backpropagation are: requires global memory which considerably complicates hardware implementations, it may converge on a local error minima and the system my oscillate or even wander.

There are many learning paradigms, and for the most part, architectures support modified versions of these paradigms. The basic models are Hebbian learning and the Delta rule [15]. The Hebb model states that if signals from other nodes are strong and the node in question has a signal then the weights should be modified so that the node in question's signal becomes stronger. A problem with Hebbian learning is overlearning or saturation of the weights. The Delta model states that the weights of the nodes should be modified to accommodate error in the output layer until the output layer is sufficiently close to the desired output.

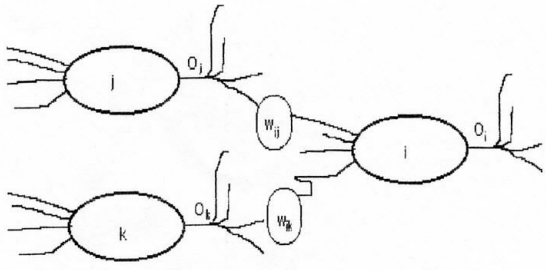

Figure 5, Self modifying analog network

As stated previously, it is too complicated to use a gradient descent architectures in hardware because of routing difficulties to access global data. What is generally done, is that a host computer samples the outputs of a hardware based system and simulates a gradient descent algorithm while in the learning mode, off-line. Another method, described in [16], a modified Hebbian learning model for local learning is implemented, see figure 5. If w is the weight, d is the delta operator and f is the weight function, the weight modification rule proposed in [16] is: $dw_{ij} = (dO_i/dO_j)f(O_i)$. The weight function maps to zero at the maxima and minima so that saturation or overlearning is eliminated. The result is a self modifying neural network implemented in hardware.

The network in [6] adopts interesting qualities from the biological network; qualities often missing from traditional neural nets. They are associative learning (classical conditioning), and nonassociative learning (habitualization and sensitization). Classical conditioning is pairing of stimuli to a response, Pavlov's dog. Habitualization is the decrease in a response by repeating the stimulus. Sensitization is enhancing the response with a strong stimulus. Instead of each synapse connecting to two neurons, each synapse connects

to three neurons. These types of networks will play an important role in building autonomous machines, "... *simple nonassociative forms of learning may be an important part of biological intelligence.*" [6]. These types of conditioning must be hardwired and therefore create expensive, dedicated systems with unchangeable topologies.

Analog Real-Time Systems

The response time of an analog system is essentially the propagation delays added to the set time constant of a layer multiplied by the number of layers and added to the time it takes for the system to settle. If r is the response time, p is the propagation delay in a layer, t is the time constant, n is the number of layers and T is settle time, then $r = (p + t)n + T$.

> For a system with 5 layers and propagation delays in each neuron of 50ns, a time constant of 10ms and a settling time of 2 time constants, the response time would be abut 70.00025ms.

Increasing the number of lateral neurons, neurons in the same layer, does not effect the response time since neurons operate in parallel. Of course, in this example, the response time does not account for the learning time, and rightly so. The programing time of a digital computer is not included in the response time and neither should the learning time. Most neural networks and in particular, the analog networks, have two modes of operation, off-line learning and on-line. By the time the system is on-line, one hopes that all scenarios have been taught and that it would do reasonably well with untaught data. There are some systems that are self modifying and in those cases the worst case convergence time would have to be added to the response time.

An analog system [4] with 72 neurons, 2,466 synapses 21,120 switches and 36 time constants with each neuron having 32 inputs was designed to decompose sound. It was found that the network converged within one time constant of the last layer. That is the value set in the time constant module. This system, although purposely slowed down (time constant), easily kept up with the spoken word. A Sun 4/110 was used to simulate the network and it was found that a speed of 100 GFLOPS would be needed to keep up with the spoken word, much faster than a Sun.

An example of process control for the fermentation of penicillin at SmithKline Beecham is shown in [1]. It was not possible to measure certain variables on-line at a rate needed to assure quality control. A complex non-linear relationship between secondary measurable variables and the process was modeled using a neural network. Because of the complex relationship, it was found to be easier to teach a neural network than to solve and program the differential equations on a digital computer. The result was a system that controlled the fermentation process in real-time.

The possibility was explored whether a neural network could be used to set parameters for flight control [10]. In this example the intended network was to be analog but the research was performed using a software implementation. The problem was that flight characteristics are non-linear and that present digital computers use look-up tables and interpolate intermediate values. It was found that the neural network was able to set parameters without any knowledge of flight characteristics. For this system to be operational, a hardware implementation would be needed. A digital system would be too large.

High-ratio image compression using an analog/digital mix [8] was performed in real-time using a VLSI neuroprocessor, NNVQ. The system was able to achieve a compression ratio of >33 and a response time of 500ns; it was 750 times faster than an equivalent system running on a Sun 3/60. This system was adaptive, and training was performed on-line. The response time for this system, therefore, includes convergence times. The upgraded version of the system was configured such that a compression ratio >20 was achieved with a response time of 250ns. The updated version increased the number of connection per se-

cond from 3.33 billion to 104 billion. The first system has an operating rate of 2MHz which is capable of operating HDTV which requires 1.2MHz. This system does not have any time constant modules, perhaps a result of the analog digital mix.

A mixed digital and analog system [7] was built to decipher handwritten characters. The first four layers were analog with the last being a digital signal processor (DSP). Of 2,000 characters tested, the network scored 5.3% in misses while human counterparts scored 2.5%. The system was able to read 1,000 cps as opposed to 20cps using a full implementation of DSP. Compare that to the Software implementation at NYNEX (see below, Software implementations of real time systems) of 1cps.

Digital Real-Time Systems

The Anza Plus [3] can perform 6,000 connection per second (C/s) on-line and 1,500 C/s while learning. The Connection machine [14] is capable 4 GFLOPS. Performance of analog systems are currently an equivalent 100 GFLOPS with systems of an equivalent 10 TFLOPS in the making. The number of connections per second for the analog systems range from 1 - 4.7 GC/s, [7].

Software Implementations Of Real-Time Systems

A system at NYNEX Science and Technology was implemented for character recognition of the dollar amount on checks. The system had to find the dollar amount on the check and decipher it. Once a single digit was located, a neural network was used to figure out what the number was. The system, overall, had a throughput of 1cps and was considered adequate for their application, although three orders of magnitude slower that the analog version described in [7].

In [4] an analog neural network was built that decomposed sound in real-time, easily keeping pace with the spoken word. A Sun 4/110 was later used to simulate the network and it was estimated that a digital computer of 100 GFLOPS would be required to match the speed of the analog system, much faster than any digital computer today

Conclusion

The design goal of the analog neural network is to replicate functions of the biological neural network because the biological network is so efficient. Major differences still exist. For the most part, except for [6], the analog systems only incorporate associative conditioning. It will be increasingly important to incorporate the nonassociative forms conditioning found in the biological systems so that, perhaps a network would be able to respond with better than a guess to unlearned inputs. Could a network be designed extrapolate from a picture of toy poodle that it was a dog, not a cat, when the only dogs taught were bulldogs and shepherds?

The number of interconnection in the biological network is far greater than the analog system. For example a biological network can have between 1,000 to 10,000 connections where the analog systems typically have 32 connection per neuron. Circuit density and routing problems limit the amount of interconnection. Contrary to the human brain, the analog network has only one type of neuron. The human brain has neurons optimized for specific tasks. The cerebral cortex, for example, has neurons with different properties than those in the vision processing center.

When is a neural network appropriate and what kind? Systems with non-linear and or fuzzy data are likely candidates for neural networks. Highly complex relationships between the inputs and the desired outputs also hold favorably for neural networks. Systems where numerical exactness may not be a requirement, the data is not well known and tremendous speed is required are likely

candidates. Some examples of these systems are: Process control, pattern recognition, light and sound decomposition, tracking systems, robot control from sensors, ignition systems and HDTV. One major problem is that, other than software simulations, neural networks are expensive. Simple analog systems cost 10^{s} of thousands of dollars. Digital systems, let alone accelerator boards such as transputers, are at least as expensive as the analog systems.

If speed is not a factor but cost is, then a software implementation might be suitable, otherwise consider an analog or digital system. If a system must have a response time within the microsecond to low millisecond range then an analog system would be the likely candidate. An analog/digital mixed architecture may be used to gain some numerical control without diminishing the response time too much. Size is also a factor; the analog systems are typically smaller than equivalent digital system. If architectural likeness to the biological neural network is required then the only choice is the analog system. Finally, familiarity with one particular system may reduce developmental time and possibly reduce costs.

Analog systems can grow very large with no degradation in performance whereas digital systems and in particular simulations will degrade in performance with size. This is due to the fact that in order for a neuron to process information, outputs from all other connected neurons must be known and therefore all neurons in the system must be accessed. In other words, they lack sufficient parallelism. Computations still retain their sequential nature, even in highly parallel digital machines. The artificial neural networks, *digital*, work in time slices covering successive layers. Software implementations calculate the state of one neuron at a time whereas the analog neural networks, neurons essentially work independently and continuously. The most striking difference between the analog and digital neural networks is that the digital system works in discrete time quanta and approach parallelism, while the analog system is continuous and parallel.

Acknowledgements

I'd like to thank James Moore of Stevens Institute of Technology for answering my many questions. I'd also like to thank Jenevieve Cerf and Robert Shapiro of NYNEX Science and Technology for an interesting look at the software implementation of neural networks.

Reference

1. M. J. Willis, G. A. Montague and A. J. Morris, "Modelling of Industrial Processes Using Artificial Neural Networks", IEEE Computing and Control, May 1992, pp. 113-117.

2. T. J. Senjnowski and P. S. Churchland, "Silicon Brains", Byte Vol 17 Number 10 Oct. 1992, pp. 137-146.

3. P. Treleaven, M. Pacheco and M. Vellasco, "VLSI Architectures for Neural Networks", IEEE Micro, Dec. 1989, pp.8-27.

4. Jan Van der Spiegel, Paul Mueller, David Blackman, Peter Chance, Christopher Donham, Ralph Etienne-Cummings and Peter Kinget, "An Analog Neural Computer with Modular Architecture for Real-Time Dynamic Computing", IEEE Journal of Solid-State Circuits, Vol 27 No. 1, Jan. 1992, pp.82-91

5. Paul Mueller, Jan Van der Spiegel, Vincent Agami, Perves Aziz, David Blackman, Peter Chance, Adnan Choudhury, Christopher Donham, Ralph Etienne, Lisa Jones, Peter Kinget, Walter Von Koch, Jinsoo Kim and Jane Xin, "Design and Performance of a prototype General Purpose Analog Neural Computer", IEEE, Jan 1991, pp.I-463I-468

6. Christian Schneider and Howard Card, "Analog CMOS Synaptic Learning Circuits Adapted from Invertebrate Biology", IEEE Transactions on Circuits and Systems, Vol 38, No 12, Dec. 1991, pp.1430-1438.

7. Bernard E. Boser, Eduard Sackinger, Jane Bromely, Yann Le Cun and Lawrence D. Jackel, "An Analog Neural Network processor with Programmable Topology", IEEE Journal of Solid-State Circuits, Vol. 26, No. 12, Dec. 1991, pp. 2017-2025.

8. Bing J. Sheu and Wai-Chi Fang, "Real-Time High-ratio Image Compression using Adaptive VLSI Neuroprocessors", IEEE July 1991, pp. 1173-1176.

9. A. Cousein, "Neural Networks for Robot Control", IEEE, pp. 119-124.

10. A. Aziz Bhatti, "Neural Network Based Control System Design of an Advanced Fighter Aircraft", IEEE Oct. 1991, pp. 1-6.

11. Paul Hasler and Lex Akers, "Implementation of Analog Neural Networks", IEEE May 1991 pp. 32-38.

12. Novat Nintunze, Angus Wu, Pichet Chintrakulchai and Jack Meador, "VLSI Implementable Classically Conditioned Neural Elements", IEEE Jan. 1990, pp. 281-282.

13. Paul Mueller, Jan Van der Spiegel, David Blackman, Timothy Chiu, Thomas Clare, Joseph Dao, Christopher Donham, Tzu-pu Hsieh and Marc Loinaz, "A General Purpose Analog Neural Computer", IEEE pp. II-177-II180.

14. Andrew S. Tanenbaum, "Structured Computer Organization", Prentice Hall Third Edition, 1990, pp. 512-520.

15. Bart Kosko, "Neural Networks and Fuzzy System", Prentice-Hall, 1992.

16. Hiroyuki Wasaki, Yoshihiko Horio and Shogo Nakamura, "A Localized Learning Rule for Analog VLSI Implementation of Neural Networks", IEEE Jan 1990.