Hybrid computation with an attractor neural network

Conference Paper · February 2002

DOI: 10.1109/COGINF.2002.1039275 · Source: IEEE Xplore

1 author:

James A. Anderson
Brown University
79 PUBLICATIONS **5,776** CITATIONS

SEE PROFILE

Hybrid Computation with an Attractor Neural Network

JAMES A. ANDERSON
Department of Cognitive and Linguistic Sciences
Brown University
Providence, RI 02912
James_Anderson@brown.edu

This paper discusses the properties of a controllable, flexible, hybrid parallel computing architecture that potentially merges pattern recognition and arithmetic**.** Humans perform integer arithmetic in a fundamentally different way than logic-based computers. Even though the human approach to arithmetic is slow and inaccurate for purely arithmetic computation, it can have substantial advantages when useful approximations ("intuition") are more valuable than high precision. Such a computational strategy may be particularly useful when computers based on nanocomponents become feasible because it offers a way to make use of the potential power of these massively parallel systems.

1. Introduction

I have been interested for a while in the problem of "hybrid" computation, that is, building a system that displays both logic based and associative computing. One ultimate goal of such a project could be a physically hybrid computer architecture combining the best properties of the classic, logic based digital computer, and a largely associative, perception based neural net computer. The human brain seems to have evolved the crude, but successful, beginnings of such a system. Artificial systems built along these lines might be of practical interest.

These notes describe early stages of what may become a controllable, flexible hybrid parallel computing architecture that combines pattern recognition and arithmetic. A version of this architecture has been developed and tested in a prototype computer program, the Neural Network Program Program [NNPP].

The nervous system treads a delicate but productive balance between continuous, analog based computation and discrete, logic based computation. Integer arithmetic is the quintessential computational function. A logic based computational device, for example, the familiar digital computer, does integer arithmetic quickly and accurately. A great deal is known about how humans perform integer arithmetic. Humans perform it slowly and badly. This paper will suggest that:

- Humans perform integer arithmetic in a fundamentally different way than logic-based computers.
- The way humans perform arithmetic can have substantial advantages when useful approximations ("intuition") are more valuable than high precision.

These results led to development of a more formal functional computer model of human arithmetic computation based on a nonlinear neural network. Such a model is not merely an academic curiosity from cognitive science. When used as the basis of a more general parallel computing architecture, it may have important practical applications:

- Human arithmetic computation can be modeled by using a neural net based dynamical system with discrete attractors corresponding to integers.
- The proposed technique used allows flexible, simple, and accurate programming of certain classes of arithmetic operations.
- This technique is slow and inaccurate for purely arithmetic computation. However, when such a controllable network is combined with the pattern recognition abilities of a neural network, a powerful fully parallel

This research was supported by DARPA Award MDA972-00-1-0026 to the Brown University Center for Advanced Materials Research.

hybrid computational architecture may emerge.

- Such a computational architecture may be particularly valuable when computers based on nanocomponents become feasible. It offers an effective way to make use of the potential power of these massively parallel systems.

2. Discrete vs. Continuous

Warren McCulloch and Walter Pitts in their famous 1943 paper, *A logical calculus of the ideas immanent in nervous activity*, suggested that "neural events and the relations among them can be treated by means of propositional logic." (McCulloch and Pitts, 1943 [1965], p. 19). They were led to this conclusion from their model of the neuron that approximated it as a binary device that computed logic functions based on its inputs. However, we now know that neurons act much more like small, sophisticated, and powerful analog computers. As Warren McCulloch himself put it a few years later in his essay, *Why the mind is in the head*, (McCulloch, 1951 [1965]), "sense organs and effectors are analogical. For example, the eye and the ear report the continuous variable of intensity by … impulses the logarithm of whose frequency approximates the intensity. But this process is carried to extremes in our appreciation of the world in pairs of opposites. As Alcmaeon, the first of experimental neurophysiologists, so well observed, 'the majority of things human are two – white black, sweet-bitter, good-bad, great-small.'"(p. 73)

If the "logic", that is, the discrete states, is not present in the neurons themselves, then where is it? A recent popular suggestion is that it is in discrete attractor states arising from an underlying continuous dynamical system. Such a system forms a hybrid computational architecture that combines valuable aspects of both discrete and continuous systems. In the next sections, we will describe how such a system is consistent with human performance on one set of arithmetic tasks and how abstractions of it can be generalized to wider, and potentially useful, domains.

3. Arithmetic in Humans

Previous work (Anderson, 1998) reviewed the cognitive aspects of elementary arithmetic fact learning. Some basic results are summarized in this section. This work used multiplication as a test system, but the basic conclusions reached seem to be more general.

When a computer multiplies two single digit integers, it performs a sequence of formal operations based on logic, the definitions of integers and the rules of binary arithmetic. Humans seem to do multiplication quite differently, using a combination of estimation and memory. The human algorithm might be formalized something like "Choose the product number (that is, a number that is the answer to *some* multiplication problem) that is about the right size." For example, the most common error for 9x6 is 56. More careful inspection indicates additional associative components, for example, errors are frequently "off by one" multiples of 6 or of 9, for example, 48 or 63.

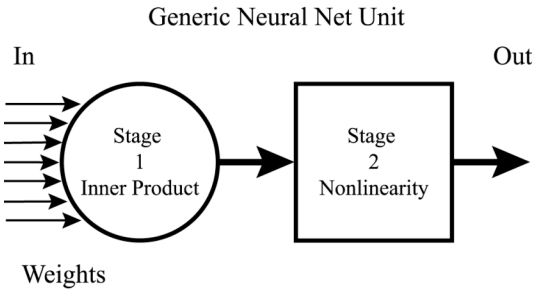
The error rate for elementary multiplication facts in college-educated adults is high, as high as 8% in some studies (Graham, 1987). Such a high rate is remarkable considering that several years are devoted to practicing arithmetic facts in elementary school, at a time when learning of other factual material -- language, for example -- is fast and accurate. However, there are also some positive aspects to such an error prone algorithm. For example, errors are usually "close" to the correct answer. Sometimes being "close" is good enough and is often much better than the kind of gross errors that malfunctioning computers are notorious for.

Attempts by cognitive scientists to model human number performance have virtually always assumed the presence of an important "perceptual" part to the internal human number representation. In many experiments, numbers act more like "weights" or "light intensities" than abstract quantities. There is also evidence from numerous sources (see Hadamard, 1949 for a classic description) that higher mathematics as actually done by mathematicians or physicists is often more perceptual than abstract. In practice, the classic theorem-proof process is primarily used to check for errors and as a way of convincing others that results are correct. These ill-defined but widely used sensory and perceptual aspects of mathematics as it is

actually practiced often go by the name "mathematical intuition."

4. A Nonlinear Neural Network

This paper is not the place to review neural network theory. Many introductions are available including my own text. (Anderson, 1995). The basic idea is that there are a large number of interconnected elementary computing units that operate in parallel. The computing units themselves are loosely based on the properties of neurons, and act as integrators of their inputs. Their connections are abstractions of synapses. The figure shows the generic two stage **neural network computing unit**. The unit



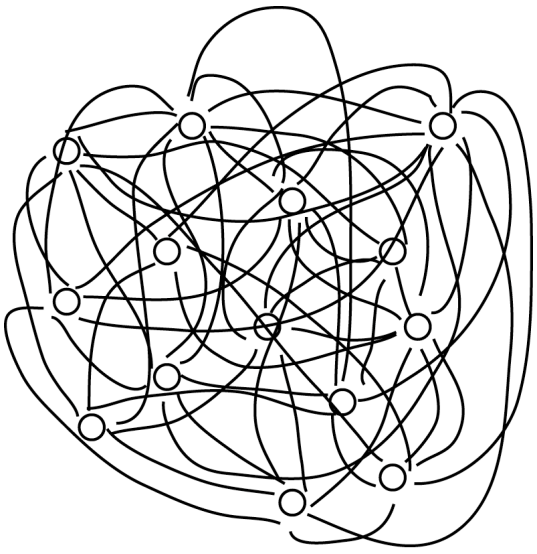
receives continuous valued inputs from other units over a potentially large number of input lines. The connections between "pre-synaptic" and "post-synaptic" units typically have continuous values. The first stage typically takes the inner product between the pattern of input activations, represented as a state vector, **f**, and the set of weights, $\mathbf{w}_i$, that is, [**f**, $\mathbf{w}_i$]. The second stage performs some nonlinear operation on the output of the first stage.

This figure of the generic neural net computing unit should be compared with the basic unit of the "Network of Networks" model of Section 7.

Neural networks form natural pattern recognizers. A complex low-level input pattern can become associated with a particular class name, for example a particular drawn input pattern can become associated with a letter. However, pattern recognition involves an implicit nonlinearity. A whole range of items – different handwritten digits, for example – is given the same output classification. It is possible to perform this classification operation several ways. The way of most interest to us involves the use of what are called **attractor networks**.

Typical attractor networks are Hopfield networks and the one we used in NNPP, the BSB network. These are both single layer **recurrent networks**, where a set of units connects to itself. In operation, the system state is started at a particular point in state space and as time progresses the system state evolves in time. It can be shown that attractor networks as they evolve are minimizing a system energy function (Lyapunov function). The system state will eventually reach a stable state in an energy minimum. The long-term behavior of these networks is characterized by these stable **attractor** states. All the points in state space that end in the same attractor are referred to as the **basin of attraction** for that attractor. Both Hopfield and BSB networks have single points as stable states, that is, **point attractors** but more complex attractor behaviors such as limit cycles are possible for other classes of recurrent networks.

Learning algorithms allow the stable states to become associated with categories or "memories." One mode of operation of such a network would be to have many different examples of the same handwritten letter, say, end in a common stable point which could then serve as the prototype pattern for an entire category. Several models for human concept formation in the cognitive literature have taken this form.



Recurrent Network:
A Layer Connects to Itself

The network used in the NNPP program is the BSB network, a one layer recurrent net with a simple limiting (i.e. clipping) non-linearity in the second stage of the units. The BSB algorithm can be analyzed easily as a feedback system. (See Anderson, 1995). If the connection matrix connecting the set of units to itself is **A**, the current state vector at time step t is $\mathbf{x}(t)$, the original input to the network is $\mathbf{x}(0)$, and the state of the system decays with a decay parameter γ (between 0 and 1), then the next state of the system, $\mathbf{x}(t+1)$ is given by

$$\mathbf{x}(t+1) = \text{LIMIT}(\alpha\mathbf{A}\mathbf{x}(t) + \gamma\mathbf{x}(t)\ \delta\mathbf{x}(0)).$$

Term weighting constants are α and δ. The qualitative behavior of the system is quite insensitive to the exact values of these parameters. The LIMIT function clips the elements of the state vector so if an element value exceeds in magnitude the upper or lower limit, the value of the element is replaced by the value of the limit. This system can be analyzed as a feedback system, with dynamics related to the "power method" used for numerical extraction of eigenvectors. The low energy points (attractors) of this system can be shown to be generally located at the "corners" of the box of limits. The neural network learning algorithms work nicely with this network to set the connection strengths. In the NNPP simulations we used the LMS (Least Mean Squares) supervised error correction algorithm.

We now can define network computation with an attractor network as the following:

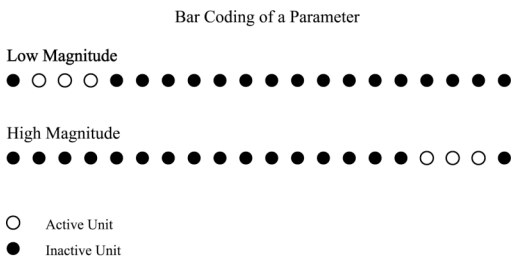
Network Computation

- **The network has built an attractor structure through previous learning.**
- **Input data combined with the program for the computation gives rise to a starting point in state space. For a neural network, the conversion of input data into a state vector is called the data representation.**
- **The network state evolves from this starting point.**
- **The final network stable state gives the answer to the computation.**

5. Data Representation for Number

The most difficult problem in the application of neural networks is converting the input data into the state vectors that can be manipulated by network dynamics. There are few explicit rules that determine what the best data representations are for a particular problem. For example, there is a constant tension between accuracy and generalization since generalization requires an "appropriate" response to an unlearned pattern. Good generalization is critically dependent on the details of the specific application and the data representation. To be useful in practice, generalization also has to be controllable so the same network can generalize one way at one time and in a different way with different task requirements

A combination of computer simulations and inference from experimental data has suggested a useful data representation for number. The starting point for our formal system will be a suggestion for the representation of number in a neural network. Topographic representations of parameters are very common in the nervous system. For example, in vertebrates, the early stages of the visual cortical representation are topographic in that the image on the retina is mapped onto the two dimensional cortex. A topographic map of the visual field is used in vision in animals as diverse as humans and the horseshoe crab, *Limulus*. There are many other topographic maps in vertebrate cortex, for example, somatosensory maps of the body surface and frequency of sound in audition.

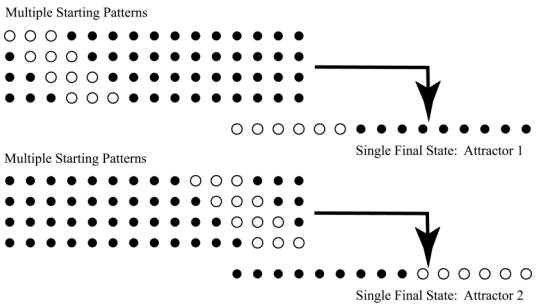


In a neural network, one form of such a topographic representation is called a **bar code**. The value of the represented parameter depends on the location of a group of active units on a linear array of units. The price paid for this data representation is inaccuracy and inefficiency in use of units. If a single parameter is represented only as a location then many units are required to

represent a value that a single unit could represent by an activity level. Such a physical mapping is also inherently low precision, with precision determined by the number of units devoted to representing the parameter. Such a representation technique, and its variants are best implemented in a system that has many relatively inexpensive units and performing a function that is only secondarily concerned with high accuracy. Nanocomponent based computing devices may fall into this category, as does, of course, the nervous system.

When used to represent data in artificial neural networks, topographic maps can be very effective for many purposes. One example is the representation of radar signal parameters used in a system to detect and analyze the structure of complex electromagnetic environments. (Anderson, Gately, Penz, and Collins, 1990)

Such a representation can represent a continuous range of values. However, suppose we couple such a representation with a nonlinear attractor neural network. It is not hard to use any of a number of learning algorithms to form a small discrete set of attractors. For example, a number of possible bar coded magnitudes can be mapped into two stable output attractors.

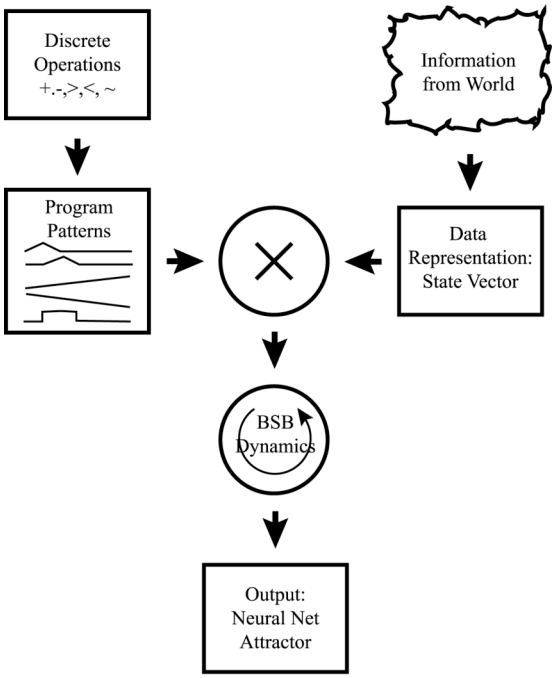


In the initial versions of a network model for arithmetic fact learning (Anderson, 1998) we used a hybrid data representation. The state vector contained two parts, one a bar code and one an arbitrary pattern that could be used to build abstract associations, for example, between the names of the numbers. For the NNPP simulations we only used the bar code part of the data representation, as the ability to perform random name associations is not needed. NNPP used bar codes for the ten digits, one through zero. We assumed that zero corresponded to 10. The bar codes overlapped, that is, the patterns were not orthogonal. This turns out to be an important coding assumption since adjacent digit representations are thereby correlated. The goal of learning is to construct a network that has ten attractors corresponding to the ten digits. This problem is not hard for standard learning algorithms. In NNPP the LMS algorithm "learns" the ten digits to a high degree of accuracy in about ten epochs, that is, ten passes through the entire set of input patterns.

6. Programming Patterns: Controlling the Computation

Learning numbers is only the beginning of arithmetic. If we want to build a useful computational system, we have to be able to control and direct the computation to give answers for specific problems. Therefore, our first job is to specify the operations that we would reasonably expect an "arithmetic" network to perform. Let us suggest several primitive candidate operations for our computing system. These operations are realized in NNPP.

Counting seems to be present in all human societies of which we have records and evidence for it goes far back into prehistory. There is good evidence that nonhuman primates and many higher mammals can count up to small integers. Formally, we can represent this function as two related operations: starting with a digit, add one to it (**increment**), and, the symmetrical operation, subtract one, that is, **decrement**. Another valuable arithmetic related



function that very small children can perform are

number comparisons, that is, the equivalent of the formal operations **greater than** or **lesser than**. Another useful operation that children can do very early is **round-off**, that is, two and a bit can be reported as "about two."

Therefore we will suggest that these five operations are a good starting point:

1. **increment (add 1)**
2. **decrement (subtract 1)**
3. **greater than (given two numbers, choose the larger)**
4. **lesser than (given two numbers, choose the smaller)**
5. **round-off to the nearest integer**

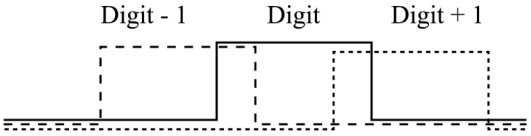
In a previous paper (Anderson, 1998), I suggested a way to control a network using a **vector programming pattern** that multiplied term by term the state vector derived from the input data. The data representation we are using – the overlapping bar codes suggested earlier – contains enough information about the relations between digits to perform these operations rather easily. The overall architecture of the system used in NNPP is presented in the figure.

A discrete operation is chosen. This operation is associated with a programming pattern. In the other branch of the computation, information from the world is represented as a bar code. These two vectors are multiplied term by term. BSB attractor dynamics are then applied. The state vector evolves to an attractor that contains the answer to the problem.

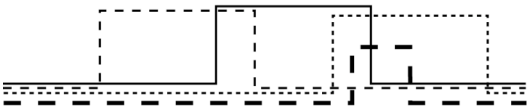
Given the data representation we have discussed, it is surprisingly easy to find programming patterns that work. This robustness arises because we are dealing with qualitative properties of the geometry of representation, that is, representational topology.

We can present intuitive arguments to support the particular programming patterns used in NNPP. Consider counting. If we start from a particular location on the topographic map, we know that one direction on the map corresponds to larger numbers, the other to smaller. Therefore, if we differentially weight the map so that nearby larger numbers are weighted more strongly, the system state can be encouraged to move toward the attractor corresponding to the next largest digit. (The heavy dashed line in the figure.)

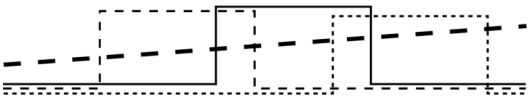
Basic digit representation: Overlapping Bars



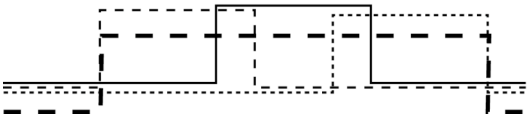
'Increment' programming pattern:



'Greater Than' programming pattern:



'Round-Off' programming pattern:



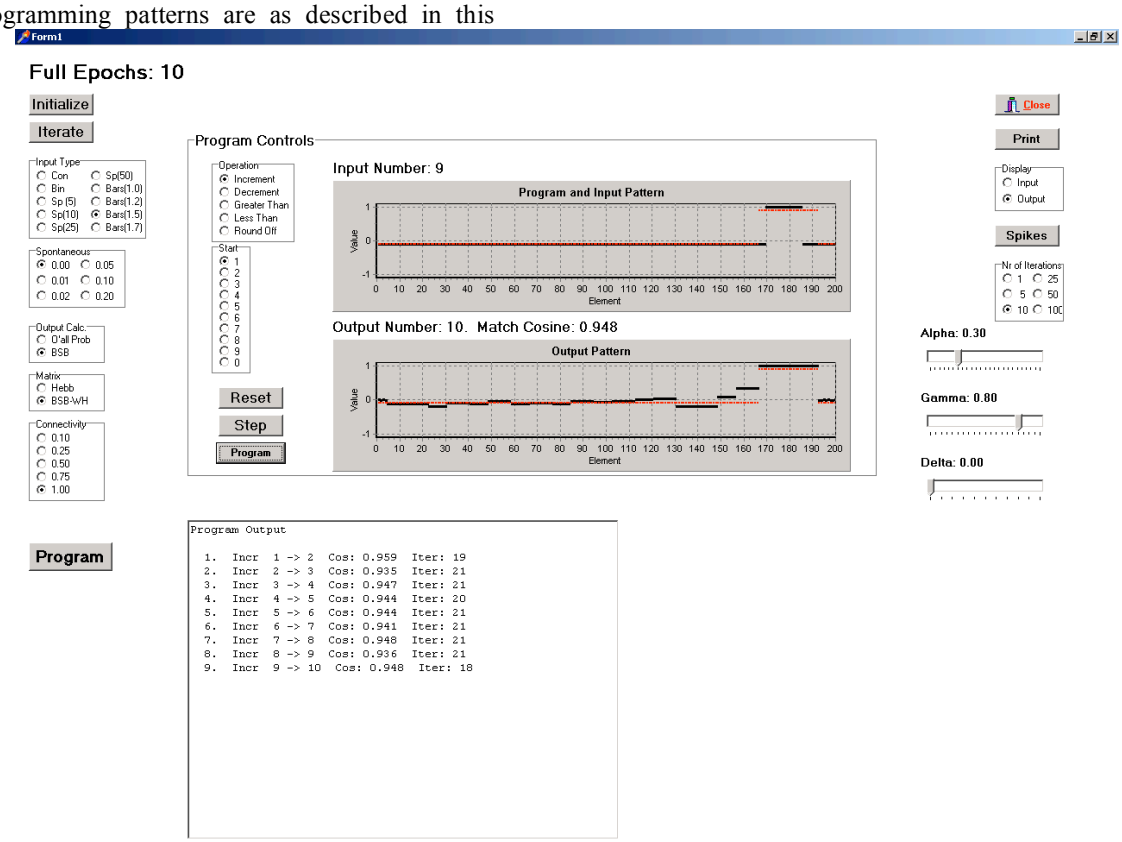
Similarly, if we have to choose which of two digits is the largest, we can do so with a similar differential weighting of the input data, so large digits reach attractors before small digits. Finding the smallest digit uses the same weighting in reverse.

Round-off is performed slightly differently. A bar for an intermediate value is generated at a location between the two digits. The network then moves to the attractor with the largest overlap with the input data bar. The figure presents sketches of the patterns that are used in the simulation. The two missing operations – less than and decrement – are reflections of the patterns for greater than and increment.

There are some peculiarities in the operation of NNPP that are worth mentioning. When the 'greater than' program runs, it displays what is called a **symbolic distance** effect in cognitive psychology. The time to an attractor is a function of the distance in magnitude between the two numbers being compared., that is, the larger of the pair [9,1] is computed faster than the larger of [5,4]. The intrusion of statistical and perceptual components into what appears at first to be a purely abstract computation is to be expected, though perhaps not welcomed. Notice

that the topographic map is effectively a realization of the number line model for integers.

The figure below presents a screen shot of NNPP running its "counting" program. NNPP had previously learned the integers, bar coded as overlapping bars in a 200 unit network. The programming patterns are as described in this section. This figure represents an actual neural net counting program in the sense that the output of one increment operation is used as the input to the next increment operation. The results of the count from 1 to 10 are presented in the box labeled "Program Output."



7. Lateral Spread: The Network of Networks

Although the programming techniques suggested in the previous section work reliably, they are appallingly slow, limited in precision and dynamic range, and are clearly of no value for constructing a practical computing system that works with precise numbers. But neural networks are excellent pattern recognizers. Even if our proposed system does not do arithmetic very well, it should be able to combine simple arithmetic operations with pattern recognition to give an example of a novel hybrid computing device.

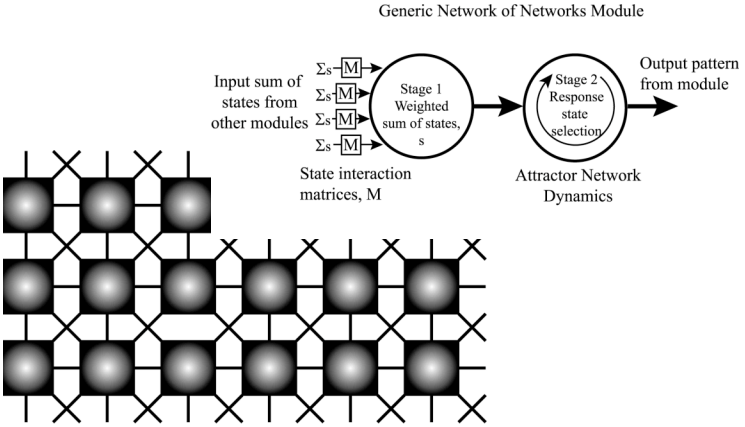
We would like to develop and investigate the properties of a controllable, flexible hybrid parallel computing architecture that merges pattern recognition, integer arithmetic, and, perhaps eventually, logic.

The computational architecture to do this computation involves both the numerical network previously presented and an additional component. A few years ago Jeff Sutton (Harvard Medical School, now NSBRI) and I proposed an *intermediate scale* neural network model we called the **network of networks.** (Anderson and Sutton, 1997) We called it intermediate scale because its basic level of organization fell between single units (1 unit) and brain regions (10^6 units.) Very little is known experimentally about what happens in this region of neural scale except that it is almost certainly both interesting and important. In behavior, the network of networks model is most closely related to the medial axis networks and

shock wave networks used in computer vision that derive, in turn, from the work of Harry Blum (1973). (See also Kimia et al, 1995, Siddiqui et al, 1999) Work by Kovacs and Julesz (1994) and Kovacs, Feher, and Julesz (1998) among others have provided experimental data supporting variants of this approach.

The network of networks architecture was based on the assumption that the elementary computing units in the nervous system are not single units as in simple neural networks, but modules composed of many interconnected single units. The modules themselves are locally interconnected with other modules and repeat in a regular structure. Modules are assumed to be complex neural networks, constructed from elementary units but to some degree removed from them since they are organized at a different level. In the brain model these elementary networks were assumed to be attractor networks, that is, neural networks realizing a nonlinear dynamical system whose behavior is dominated by attractor states. A state in one module influences the state of an adjacent module by influencing the weights of the activities of the attractor states. In the approximation we used, the modules had a linear region, where inputs from different modules added linearly forming a superposition of the module's attractor states and a nonlinear region where the module chose a single attractor state from the set of attractors. The temporal evolution of a module is from a sum of potential attractors initially to a single attractor finally. Notice the similarity to quantum computing in this property, though the number of considered alternatives is vastly less.

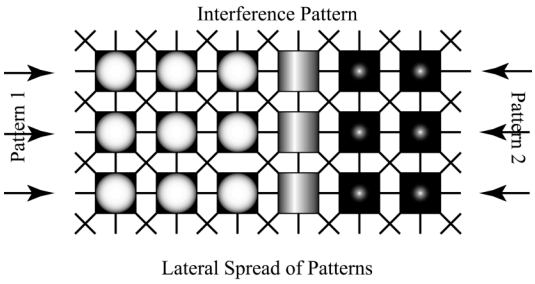
The use of intermediate scale structure has a number of advantages from the point of view of building computational devices based on nanocomponents. Intermediate level modules may be big enough to work with and control, even in a system built from nanostructures. This structure may also offer some intriguing computational approaches to difficult problems.



etwork of Networks Modular Architecture

In the original description of the network of networks model, an attractor neural network, for example a Hopfield net or the BSB model, was used to form the elementary module. This meant that the resulting network had multiple stable states in the form of point attractors. Therefore the behavior of a system with many elementary units was dominated by a much smaller number of attractor states. The network behavior was largely buffered from the specific properties of single units.

Connections between the modules are assumed to be primarily local, between nearby modules. Local connections allow the number of connections to scale roughly linearly with the number of modules rather than as the square. Because connections are local, much information processing takes place by movement of information laterally from module to module. This lateral information flow requires time and some important assumptions about the initial wiring of the modules. It is not well understood yet by us though there is currently considerable

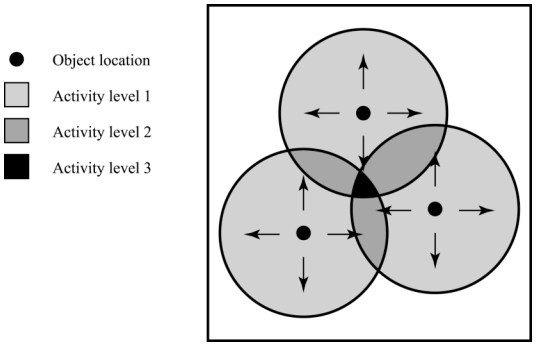


experimental data supporting the idea of lateral information transfer in cortex over long distances. The lateral information flow allows the potential for the formation of the feature combinations in the interference patterns, very useful for pattern recognition.

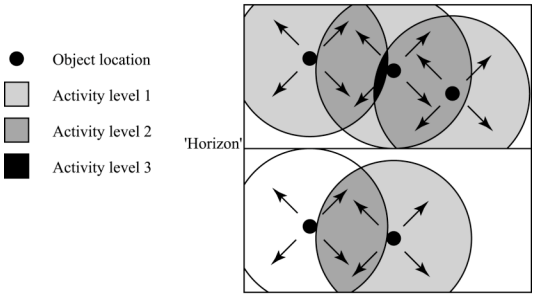
8. Combining Pattern Recognition with Discrete Operations.

Let us try to find a place where we can bring together the simple 'abstract' structures we have developed and a network architecture such as the network of networks and see what we can do. One simple test vision problem might be the following. Given a set of identical items presented in a visual field, count how many there are. There is a surprisingly large amount of research on this specific problem and its variants

in cognitive psychology and visual perception. There are several reasons for this. Experimental results suggest determining the number of objects present in a visual field proceed by what are conjectured to be fundamentally different means. From one to about four items determination of number proceeds in what is called the **subitizing region**. In this region subjects "know" quickly and effortlessly how many objects are present. Each additional item (up to 4) adds about 40 msec to the response time. In the **counting region** (beyond 4 objects in the field) each additional item adds around 300 msec per item, a figure consistent with other tasks where explicit counting is required. There is brain imaging evidence that different neural mechanisms are involved in subitizing and counting. (Sathian et al., 1999) Interestingly, this study suggests that there is a single subitizing brain region in the occipital extrastriate cortex whereas explicit counting involves widespread brain activation. Our model uses a single visual region.



Our basic computational idea is very simple. The network of networks model propagates pattern information laterally. If a number of identical objects are present, they will all be propagating the same pattern information, that is, the same features and combinations of features. Our attractor networks are linear for small signals. Therefore, let us assume we are operating in the linear region of modular interactions, that is, the individual modules have not yet moved near attractor states. Let us assume that when two pattern waves from different sources arrive at the same location they add. Patterns from identical features will add amplitudes linearly; patterns from different features can interfere. After sufficient lateral spread has occurred, the ratio of the maximum activation of a given feature or set of features compared to the initial activation, will give the number of objects. An integer can be produced the output is processed by the round-off operator of the numerical network.



Because of the non-linearity of the modules, several computational extensions are obvious. It is possible to segment the field by using modules in attractor states. There are a number of effects (metacontrast) suggesting lateral interactions can be halted by interposing lines or regions. For example, segmentation using a line lets the system answer questions like that posed in the figure: Which grouping has more components? The one above or below the "horizon?" A more homely version of this problem might be **"Which plate has the most cookies?"** a general problem of genuine cognitive and behavioral importance.

Why is human subitizing apparently limited to four items? From our point of view there are two interesting reasons. First, the numerical count network ceases to be sufficiently accurate. Second, nonlinear effects occur in the spread of lateral activity. **It would not be difficult to go far beyond human subitizing performance in an artificial system.**

9. Future Development: Linking Computation and Pattern Recognition.

Current digital computer technology is exceedingly fast when performing logic and arithmetic. Massively parallel systems are fast and effective for some kinds of pattern recognition and vision problems because their intrinsic parallelism matches the problem. Note, however, that one architecture is bad at what the other does well. This fact presents opportunities for hybrid computer architectures that combine logic based and parallel systems.

The centrality of data representation for neural net operation has been known for a long time. However, there is little guidance beyond experience and intuition as to how to choose the

most effective representation for a given problem. The hybrid system described here, where clear discrete computational functions are to be carried out in conjunction with a traditional network immediately suggests why some representations are effective. For example, the "number line" analogy for the integers is directly realized in NNPP and forms the basis of the elementary arithmetic operations performed. Examples of some of the peculiarities that occur in arithmetic when the line analogy is not available, as it was not for much of human history, can be found in Yeldham (1926).

However the most interesting potential extension of NNPP is also the most ill formed. We have suggested that it is possible to develop a distributed, largely parallel computer architecture based on a system that harnesses the abilities of discrete and continuous systems that work together. The most important long term result of this project will be to get some formal idea of the power of this approach as an alternate or, more likely, supplemental computer architecture.

By necessity of its parallelism, speed, and the unreliability and difficulty of controlling single computing elements, a nanocomponent based computer would be an efficient computer for a different set of computational operations than a traditional computer. A major point of this paper is to give a detailed example of what such software might look like.

References

JA Anderson (1993). The BSB network. Pp. 77-103 in MH Hassoun (Ed.), *Associative Neural Networks*, New York, NY: Oxford University Press.

JA Anderson (1995). *An Introduction to Neural Networks*. Cambridge, MA: MIT Press.

JA Anderson (1998). Learning arithmetic with a neural network. Pp. 255-300 in D. Scarborough and S. Sternberg (Eds.) in *An Invitation to Cognitive Science, Volume 4: Methods, Models and Conceptual Issues*. Cambridge, MA: MIT Press.

JA Anderson, MT Gately, PA Penz, and DR Collins (1990). Radar signal categorization using a neural network. *Proceedings of the IEEE*, **78**, 1646-1657.

JA Anderson and JP Sutton (1997). If we compute faster, do we understand better? *Behavior Research Methods, Instruments, and Computers*, **29**, 67-77.

HJ Blum (1973). Biological shape and visual science (Part I*). Journal of Theoretical Biology,* **38** 205-87.

R Gelman and CR Gallistel. (1986). *The Child's Understanding of Number*, Cambridge, MA: MIT Press

DJ Graham (1987). An associative retrieval model of arithmetic memory: How children learn to multiply. In JA Sloboda and D Rogers (Eds.), *Cognitive Processes in Mathematics,* pp. 123-141. Oxford: Oxford University Press.

J Hadamard (1949). *The psychology of invention in the mathematical field*. New York, NY: Dover.

B Kimia, A Tannenbaum and SW Zucker (1995). Shapes, shocks and deformations {I}: The Components of Shape and the Reaction-Diffusion Space, *International Journal of Computer Vision*, **15**, 189-224.

I Kovacs and B Julesz (1994). Perceptual sensitivity maps within globally defined shapes. *Nature*, **370**, 644-6.

I Kovacs, A Feher and B Julesz (1998). Medial point description of shape: a representation for action coding and its psychophysical correlates. *Vision Research*, **38**, 2323-2333.

TS Lee, D Mumford, R Romero, and VAF Lamme (1998), The role of primary visual cortex in higher level vision. *Vision Research*, **38,** 2429-2454.

S Link (1990). Modeling imageless thought: The relative judgment theory of numerical comparisons. *Journal of Mathematical Psychology*, **34,** 2-41.

WS McCulloch (1951/1967). What is a number that a man may know it, and a man that he may know a number. Reprinted in WS McCulloch (Ed). *Embodiments of Mind*. Cambridge, pp. 1-18. MA: MIT Press.

WS McCulloch and W Pitts (1943/1967). A logical calculus of the ideas immanent in nervous activity. . Reprinted in WS McCulloch (Ed) *Embodiments of Mind*. Pp. 19-39. Cambridge, MA: MIT Press.

K Sathian, TJ Simon, S Peterson, GA Patel, JM Hoffman, and ST Grafton (1999). Neural evidence linking visual object enumeration and attention. *Journal of Cognitive Neuroscience***, 11**, 36-51.

K Siddiqi, B Kimia A Tannenbaum and S Zucker (1999). Shocks, shapes, and wiggles. *Image and Vision Computing*. **17**, 365-373.

FA Yeldham (1926). *The story of reckoning in the Middle Ages*. London: Harrup.