ROBERT E. JENKINS

NEURODYNAMIC COMPUTING

For several years, the Computing Science and Technology Group (S1A) of the APL Space Department has been engaged in research and development in the emerging technology of connectionist, or neural-network, systems. Some of the earlier work involved the development of the Hopkins cellular logic processor. This article describes the viewpoint on this technology that has emerged during a cooperative effort between APL and The Johns Hopkins University Department of Electrical and Computer Engineering. A number of ongoing projects within that cooperative effort are discussed.

CONNECTIONIST ARCHITECTURES

Neural-network computing technology is currently in vogue and is going through an explosion of activity that is typical of a rapidly emerging technology. The subject is not new, though, and has been an active field of research for a long time. APL is a relative newcomer in the field. It would be impossible in a paper of this scope to survey and cite all of the important past research results. Instead, we draw the reader's attention to the recent two-volume work by Rumelhart and McClelland et al.,[1] which presents most of the main ideas of the field and contains many references.

Why the sudden surge of activity in the 1980s? It is probably the result of understanding and many ideas finally coming together. A key contributing factor is the willingness of researchers (typified by J. J. Hopfield[2]) to unshackle themselves from the details of neurological systems and study physical dynamic systems that exhibit only the major architectural features of the brain. That has led to some colorful and inspirational results that captured the fancy of many people.

Neurological systems are dynamic networks of communicating cells that influence one another's behavior, and physical systems that capture that property are commonly called connectionist systems. The main architectural features of such dynamic systems are as follows:

1. The system is made up of a large number of simple, identical elements, variously referred to in the literature as units, neural elements, or automata.
2. Each unit stores very little information internally—typically only its own scalar state or "activity level."
3. Each unit is connected to some number of other units to form a network; the units transmit their activity states one to another and use that information to compute new states of activity. (The analogous connections in biological systems are called synapses.)
4. Information being processed by the system is represented by globally distributed patterns in the activity levels of large groups of units. The program for processing information (or knowledge) is stored within the system as globally distributed patterns of connection strengths, or "synaptic weights."
5. Each unit determines its new state via a nonlinear function of its summed, weighted inputs from the other units connected to it. This nonlinearity seems to be crucial to much of the interesting behavior of such systems. Earlier studies of linear connnectionist systems failed to produce some of the rich results currently observed with nonlinear models.

The idea of storing information and knowledge as distributed patterns of weights and activity over large sets of units, rather than in the processing units themselves, is a principal feature. That, coupled with the fact that each unit computes its own change in state from locally derived information (its input connections), allows the system dynamic behavior to be massively parallel in nature. Literally all of the information in the system is being processed simultaneously as the system evolves in time. In addition, for large numbers of connections, the system exhibits a great deal of fault tolerance, since failures of individual units or connections do not significantly change the global response or resulting patterns in the system state. The accepted term for computations that use such a connectionist system is "parallel, distributed processing," where the adjective *distributed* refers to how information is stored, not the spatial arrangement of the units.

Systems with the above properties are being researched from a broad range of perspectives: cognitive science, neurology/neuro-anatomy, physics/biophysics, computing science, and engineering. Our current effort, with Dr. Fernando Pineda as the Principal Investigator, has been supported by both APL internal research and development funds and a three-year grant from the U.S. Air Force Office of Scientific Research. The activity has profited from collaboration with the Homewood Campus of The Johns Hopkins University, principally Dr. Andreus Andreou of the Electrical and Computer Engineering Department and Ben Yuhas, a Ph.D. candidate in that department. Our group focus is from the computing and electrical engineering viewpoints, from

which we see two important interest areas for those disciplines.

First, neural-network models provide an interesting signal-processing tool, of which adaptive linear arrays is a special case. This tool will augment traditional methods for complex processing tasks on a wide range of signals such as grey-level images, acoustic signatures, and radar returns. The image-processing power of cellular-logic operations (a particularly simple connectionist system) is a convincing example.

Second, we advocate the (not original) viewpoint that neural networks constitute a new paradigm for computing, which will surely impact the underlying architectural structures and hardware that will implement it.

The traditional view of a computer is a finite-state machine performing sequential sets of instructions. Parallelism, through the introduction of multiple such machines, does not change that traditional view very much. On the other hand, the neural computation paradigm is a more general view of the computer as a dynamic system evolving under the control of dynamic laws and external stimulus (that is, inputs). Much of the recent excitement in neural networks was, in fact, generated by Hopfield's suggestion in 1983 that the collective properties of physical dynamic systems could be used to directly implement computing tasks. The major benefits of such a computing paradigm will be realized by hardware in which the physics of the underlying machine and the computational algorithm are intimately related.

An appreciation of the potential benefit of such an approach to computing has focused our activities onto dynamic systems with connectionist architecture that have continuous-valued states and equations of motion expressible as differential equations. Such systems have the benefits of being mathematically familiar to physicists and of relating well to analog electronics for implementation. In the past year, Fernando Pineda of APL has developed computational algorithms for a fairly general and well-known form of such systems, and has shown that those algorithms are generalizations of some earlier results.[3,4] In the following sections, that particular dynamic system is briefly described, along with its use in computational tasks. The description paraphrases and somewhat oversimplifies the theoretical material presented in Ref. 4, and it also discusses concepts that are more or less standard at this point.

Some of our earlier efforts involved cellular logic operations—simple two-state dynamic systems with connections between nearest neighbors. Our current concentration on the particular system described below by no means implies that it is the only fruitful connectionist approach. It is just that we are forced to narrow our focus because of the limitations of our resources. Further, we feel that the major payoff of such a technology will come through eventual implementations in analog microelectronics, for which the system below is particularly well suited. Such is the subtle unifying theme of the work described in this article. In the remaining sections, several projects are described that will appear to be rather loosely coupled, partly because they are of necessity funded from diverse sources within the APL Space Department, and partly because they reflect the diverse interests of the people involved in the collaboration.

A NEURODYNAMIC SYSTEM

The basic model of the system that Pineda has studied is shown in Fig. 1. The units are arbitrarily connected, and an arbitrary subset receive inputs I_i from the external world. The state of the ith unit is described by the continuous variable X_i that represents the components of the system state vector $\mathbf{X}$. The connection strengths are described by the continuous variables W_{jk} that form a connection matrix $\mathbf{W}$. The dynamic equations of the system are:

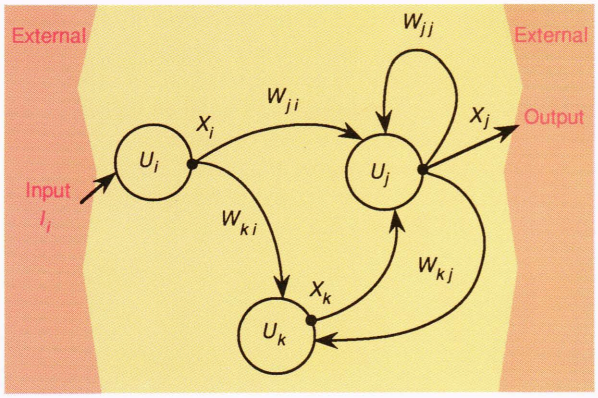
$$\frac{dX_i}{dt} = -X_i + F(U_i) + I_i \ , \qquad (1)$$

where $U_i = \Sigma_j W_{ij} X_j$. The quantity U_i is loosely analogous to the so-called "activity potential" in biological systems. The nonlinear activation function F is frequently chosen to be the continuous sigmoid given by

$$F(U) = \frac{1}{1 + e^{-U}} \ . \qquad (2)$$

The saturation property of F bounds its values between 0 and 1, and the continuous valued input vector $\mathbf{I}$ is usually normalized to that range.

Equation 1 and various forms similar to it have been widely studied in the literature. Sometimes I_i is placed



1. Each unit has an activity potential $U_i = \sum_j W_{ij} X_j$.
2. Each produces an output X_i that is a nonliner function of its activity potential: $X_i = F(U_i)$.
3. The set of X_i is a vector of real numbers specifying the state of the system at any time.
4. The network connectivity is described by a "synaptic weight" matrix W_{ij} that represents the connection strengths.

Figure 1—A dynamic connectionist network.

inside the function F and added to U_i. In some cases the models have been discrete, whereby X_i is restricted to a finite set of states, such as 0 or 1. In some cases the dynamics are stochastic. For example, rather than a time derivative, the right-hand side of Eq. 1 could be interpreted as a probability of change in the state of unit i during the next time interval.[1] In addition to Eq. 1, Grossberg[5] has over the years treated numerous similar systems, some of which are much more complex, and his work merits careful attention.

Within APL's Computing Science and Technology Group, it is generally felt that a system described by Eqs. 1 and 2 captures the major essence of neurodynamic computation. It contains, as special cases, both the continuous Hopfield Net[2] (symmetric weight matrix) and the popular feed-forward architecture[1] (lower triangular weight matrix). Also, as Hopfield has pointed out, it is amenable to implementation in analog electronic hardware. Consequently, we have concentrated on such a system, reviewing previously published work and working to contribute to the further understanding of potential applications.

When used in a processing task, the "answers" that the network gives are generally the final state-vector response to a set of external inputs or to a set of initial conditions. By final response, we mean the stable points of static equilibrium of the system. Hence, the question of stability is of great importance and is of major concern to researchers.

The general problem of stability of nonlinear, multidimensional systems is difficult; a saving grace is that, empirically, we have observed large networks described by Eq. 1 to be stable under a wide variety of connectivity patterns. It appears that one must make a definite effort to construct weight matrixes that induce instability. In fact, Atiya[6] has shown that systems described by Eq. 1 will converge to a unique fixed-point attractor if the magnitudes of the weights are small enough. The stability question is an active area of research, but for our present purposes we assume that we are dealing with a dissipative system. That is, for a fixed set of inputs **I** and initial conditions, $\mathbf{X}_0$,

$$\frac{d\mathbf{X}}{dt} \rightarrow 0 , \qquad \text{as } t \rightarrow \infty . \tag{3}$$

Under this condition, the system state **X** converges to point attractors **A** that are solutions to the equation

$$A_i = F\left(\sum_j W_{ij} A_j \right) + I_i . \tag{4}$$

We can arbitrarily denote a subset of the final state as being the network "outputs," which leads to a picture such as Fig. 2. The network can be thought of as a "black box" that responds with transient behavior when it receives a new set of fixed inputs, or when it is perturbed into a new state. Eventually the system restabilizes at a new fixed point and produces a new set of outputs.

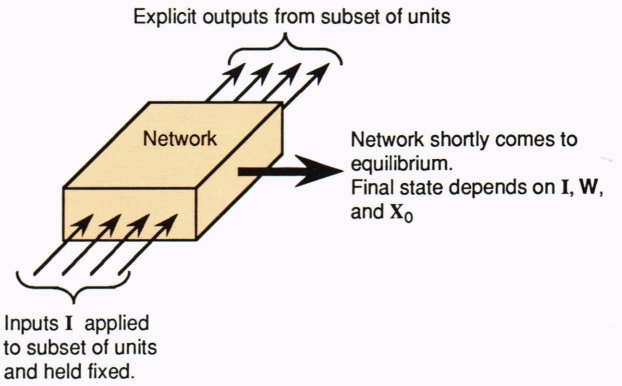


Figure 2—Dynamic neural network as a black box. Transient activity is generated each time the system is perturbed by the external world. Useful computations are done as the system comes back to static equilibrium.

LEARNING

To be useful as a computational tool, the network needs to be programmed; that is, the fixed points of the system in response to inputs must be moved around to useful spots in the state space of **A**. That is done by adjusting the connection weights, since Eq. 4 shows that the system fixed points are obviously a function of **W**. When done via an adaptive procedure, such a process is called "learning."

Computer models of neural networks learn by adjusting themselves (via some algorithm) to a sequence of external inputs, not by being reprogrammed with a new set of logic or rules. The program logic is simply the solution to the dynamics of Eq. 1, along with an algorithm for weight adjustment.

Learning is generally done through an iterative procedure that slowly adjusts the weights by small increments in a sequence of repeated presentations of patterns to be learned. For each presentation the system is allowed to stabilize to a fixed point, which is then used to determine a small adjustment to the weights. (For some special cases, such as the Hopfield Net[2] or linear networks, the weights can be computed in closed form from the patterns.)

Thus, there are three different time constants associated with the overall dynamics when learning is included. The changes in the inputs happen slowly relative to the settling time of the network, so the network stays in a kind of quasi-equilibrium with the inputs. Learning, or adjusting the weights in response to the changing inputs, occurs over time scales that are longer yet. The process is analogous to those processes that physicists call adiabatic.

There are numerous choices for learning algorithms, and it is beyond the scope of this article to review the field. The class of learning algorithm that has been the focus of our current effort is known as "supervised" learning, in which *a priori* knowledge about the desired system output is used to adjust the weights. That is, the fixed points are purposefully moved and positioned in the state space of the system. By contrast, unsupervised learning allows the fixed points to move where they may

under the weight-adjustment algorithm. In both classes of learning, the network gradually begins to respond in an organized way to temporal variations in the inputs.

One way to position the fixed points is to minimize a measure of the error in the system outputs over a range of inputs. Such an approach has become fairly standard in the literature, and has been the recent focus of our work. Consider a set α of input vectors $\mathbf{I}^{\alpha}$, on which the network is to be trained. We choose a subset k of units to represent the system output, and for each input vector in the training set there is a corresponding desired target state $\mathbf{T}_k^{\alpha}$ for the outputs. The weights can be adjusted by minimizing the positive definite function

$$E = \tfrac{1}{2} \sum_{\alpha} \sum_{k} (T_k^{\alpha} - A_k^{\alpha})^2 , \qquad (5)$$

where the vector $\mathbf{A}^{\alpha}$ is the fixed point that corresponds to the αth input pattern.

The minimization of E can be done in various ways, most of which are essentially iterative using the gradient of E in the weight space. For each individual pattern in the summation over the training set, the contribution to the gradient can be shown to be the outer product of an error term and the corresponding fixed-point state vector of the system. Pineda[4] has derived a general formulation of the error term for arbitrarily connected networks that has the form of a differential equation. The learning process in terms of the gradient is typically also cast in the form of a differential equation:[1]

$$\frac{d\mathbf{W}}{dt} = -\frac{1}{\tau} \nabla E , \qquad (6)$$

where ∇ is the gradient with respect to the components of the weight matrix $\mathbf{W}$. The parameter τ is sometimes called the gain, but is best thought of as a time constant. Equation 6 can also be made second-order in time, which is equivalent to adding what many researchers have referred to as a "momentum term."[1] That term appears in the second-order difference equation as a term proportional to the first difference in $\mathbf{W}$, and has been found to improve convergence of the steepest-descent method of minimizing E by reducing the sensitivity to fine-scale undulations of the surface.

For historical reasons, such a process is called back-propagation, and special cases of it have been used for some time.[1] The name is derived from the concept that the process propagates the observed error in the output units back into the network to correct the weights. Pineda's formulation of back-propagation as being completely described by a set of first-order differential equations includes the learning process as part of the continuous dynamic system. That is an important step toward implementing the entire system, including learning, in analog hardware, although it is not clear to us at this point how to accomplish it with electronic devices. It is also true that minimizing the function in Eq. 5 may not necessarily be the best way to accomplish learning, although it is a useful place to start for analog implementation.

PATTERN-MAPPING AND AUTO-ASSOCIATION

There are two fundamental computational tasks that can be performed by a system such as Eq. 1: continuous pattern-mapping and auto-association. Pineda's learning algorithm can be applied to either, although there are subtle differences between the two.

It is easy to see from Eqs. 1 and 4 that the fixed-point attractors of the system are a function of the weights, the inputs, and the initial conditions for Eq. 1; that is,

$$\mathbf{A} = g(\mathbf{W}, \mathbf{I}, \mathbf{X}_0) . \qquad (7)$$

For a given connection matrix and a fixed input vector, there may be more than one fixed-point attractor, and if the system starts in a state nearby one of these, it will stablize at that point. Use of the term *nearby* implies a basin of attraction associated with each fixed point. If the weight magnitudes are small enough (Atiya's condition[6]), the fixed points are degenerate, and all possible initial conditions lie in a single basin of attraction associated with a unique fixed point. Pattern-mapping makes use of that property, although the uniqueness properties are not well understood for networks with feedback connections.

In pattern mapping, the value of $\mathbf{X}_0$ is set to a convenient fixed value (such as all zeros), an input is applied, and the system is then allowed to relax to its fixed point of equilibrium. Provided that Atiya's condition is satisfied, that establishes a unique, stable output state. If the input is changed an infinitesimal amount, the output state will correspondingly change because F in Eq. 4 is a continuous function. The output thus becomes a continuous, but complicated, function of the input. In that case the system will "interpolate" in an extremely complex function if it has been programmed through learning to give the correct output for a sufficient number of cases that span the input domain. Pictorially, the process could be visualized as shown in Fig. 3.

As an example of that type of computation, suppose $\mathbf{I}^{\alpha}$ is a set of 3000 or so carefully selected backgammon situations, that is, suitably encoded representations of the markers at each point and the current dice throw. The required corresponding outputs $\mathbf{T}^{\alpha}$ would be suitably encoded representations of appropriate moves. After adjusting the weights so that all the situations in the training set give the correct output, the network has presumably formed some internal representation of the function relating the appropriate move to the board position. The network then plays a reasonable game of backgammon by interpolating that function for new positions. This example is a simplified description of actual work done by G. Tesauro and T. Sejnowski using the back-propagation learning algorithm on networks of about 500 units.[7]

In the auto-associative mode, the patterns to be learned are stored in the network as retrievable memories rather than input mappings. In this case, the value of the input vector $\mathbf{I}$ is set to a fixed value (usually 0). Then

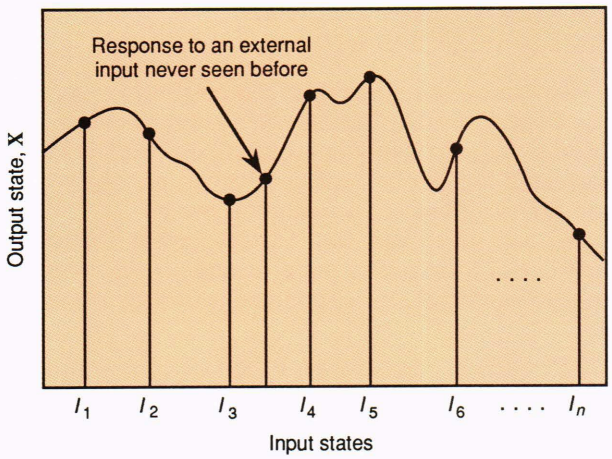


Figure 3—Pictorial view of pattern-mapping mode. Training set $I_1 - I_n$ is used to adjust the synaptic weights to force the network to form an internal representation of a complicated function. Since the output state is a continuous function of the input, the network will interpolate in response to a new input that is not a member of the training set.

an initial condition $\mathbf{X}_0$ is applied and the system is allowed to converge to the fixed point within whose basin of attraction the initial state lay. The state of the system at that point becomes the output. In this mode, the learning process proposed by Pineda[4] breaks the degeneracy in the fixed points and moves them to positions corresponding to the patterns to be stored, so that the network behaves in the same manner as an associative memory. The training set in this mode becomes the set of patterns used as initial conditions $\mathbf{X}_0^{\alpha}$ rather than as inputs. Unlike the pattern-mapping mode, the output of the system is now a sharply discontinuous function of the initial conditions.

As an example of such a computation type, suppose $\mathbf{X}_0^{\alpha}$ is a set of normalized grey-level images of faces. The learning process would establish each image as a fixed-point attractor for the system. If a noisy or distorted version of one of the faces is used as the starting state, the system will stabilize at the fixed point corresponding to that face. The initial state need only lie in the correct basin of attraction. This pattern-recognition example has been used by our group to test the auto-associative learning mode of Eq. 6. Some of the results are shown in the next section.

The two computations are interesting approaches to function-mapping and pattern-recognition because the rules or underlying algorithms do not have to be explicitly expressed or even known. In both cases, the networks are programmed with a set of examples, which can be a great advantage in certain problems. Of course, it is quite possible that a completely different and more traditional approach could be used to develop a computer algorithm that would be programmed with examples. We stress again our view that the main advantage of an approach based on dynamic systems (Eq. 1) is the potential for direct implementation in analog electronics.

APPLICATION TO OPTIMIZATION TASKS

There is another kind of computation that can be performed by a system represented by Eq. 1, which derives from the property of global stability. If the system is in fact asymptotically stable, then, by definition, there will exist what is referred to as a Lyapunov function for the system. That is, there will exist at least one positive definite function of the weights, inputs, and state vector that will monotonically decrease to a global minimum as the system stabilizes to a fixed point from any starting condition. That is a (nonrigorous) statement of the second theorem of Lyapunov.

Let us suppose for the moment that we could find such a function, denoted as $L(\mathbf{X},\mathbf{W},\mathbf{I})$. Let us further suppose that some optimization problem can be explicitly formulated in terms of minimizing a cost function, $C(\mathbf{P},\alpha,\beta)$ with respect to a set of parameters, $\mathbf{P}$. Suppose now we can perform a direct mapping of the cost function to the Lyapunov function:

$$C(\mathbf{P},\alpha,\beta) \leftrightarrow L(\mathbf{X},\mathbf{W},\mathbf{I}) \ , \tag{8}$$

by making a functional correspondence between $\mathbf{P}$ and $\mathbf{X}$, and between the sets of constants α, β and $\mathbf{W},\mathbf{I}$. Then, having set $\mathbf{W}$ and $\mathbf{I}$ to the values corresponding to α and β, when the system stabilizes from any starting condition, the final state vector $\mathbf{X}$ is the solution to the minimization problem.

For the special case of a symmetric weight matrix, Hopfield[8] was able to derive for Eq. 1 a Lyapunov function that happens to be quadratic in $\mathbf{X}$. He showed how that function could be mapped to the traveling salesman problem, and demonstrated near-optimal solutions obtained by simulations of the dynamic system. The traveling salesman problem is one of a class of difficult computational problems in computer science. Since Hopfield's work, there has been a plethora of published applications of this technique to various similar optimization problems, such as resource allocation, optimal receivers, and tree search.[9]

To my knowledge, Hopfield's result for the symmetric case is the only known Lyapunov function for the system described by Eq. 1. Other less restrictive conditions for global stability may exist that could lead to solutions to other optimization problems. In fact, there is no reason to restrict the method to Eq. 1. The general technique of solving optimization problems by simulating an appropriate dynamic system may be a viable algorithm for existing computers. The question is whether the required computation time for the simulation would be an improvement over existing search algorithms. I feel that such work could lead to interesting research possibilities for computer scientists who dare to stray from conventional approaches. An exciting prospect, of course, is to have analog systems with programmed weights that will converge to solutions in milliseconds.

NUMERICAL STUDIES

To support our studies of neurodynamic computing and to investigate potential applications, we have imple-

mented fairly general software simulations of the system described by Eqs. 1 through 6. To achieve the flexibility needed for exploratory numerical experimentation, the software is written in the APL language. The programs run on IBM personal computers and on an Analogic back-end array processor for large networks.

The major computing time is spent on the matrix multiplication term in Eq. 1, and the major storage/memory problem is in holding the weight matrix **W**, which can be quite large and sparse for typical networks. To alleviate such problems, we have used a sparse matrix formulation that compresses **W** with no assumptions about its block structure, and which pipelines very well on an array processor. To improve the computation on the personal computers, the matrix multiply is written in assembly language and the extended memory space is used to hold **W**. On the Analogic machine the programs can handle networks with up to 100,000 connections, and on the personal computer runs have been made with as many as 20,000 connections.

As an example of our numerical experiments, the software has been used to test the properties of the auto-associative operating mode using digitized video images of faces as the patterns. Working with faces is fun, and they are a convenient unclassified problem domain, typifying the kind of difficult pattern-recognition that people do well but for which it is hard to quantify rules. Using the network connectivity shown in Fig. 4, six faces were learned. Tests were then performed to get a qualitative measure of the basins of attraction. The results of several of those tests are shown in Fig. 5, which shows time histories of the system state vector starting from various initial conditions.

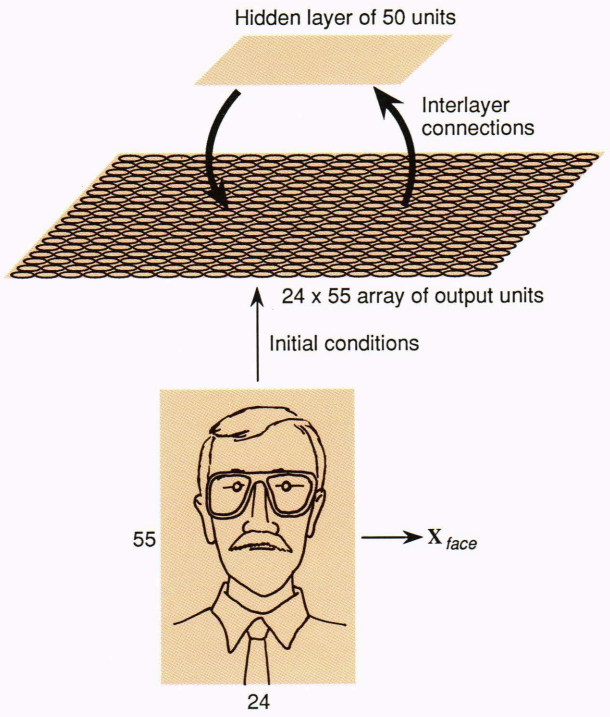


Figure 4—A network topology for the auto-associative mode. $\mathbf{X}_{face}$ is made a fixed-point attractor of the system. Any initial conditions for the units in the bottom layer that are in the basin of attraction of $\mathbf{X}_{face}$ will cause the system to recall $\mathbf{X}_{face}$.

The most important result of the tests was the ability of the system to recognize the patterns from initial conditions that were spatially scaled, translated, or rotated

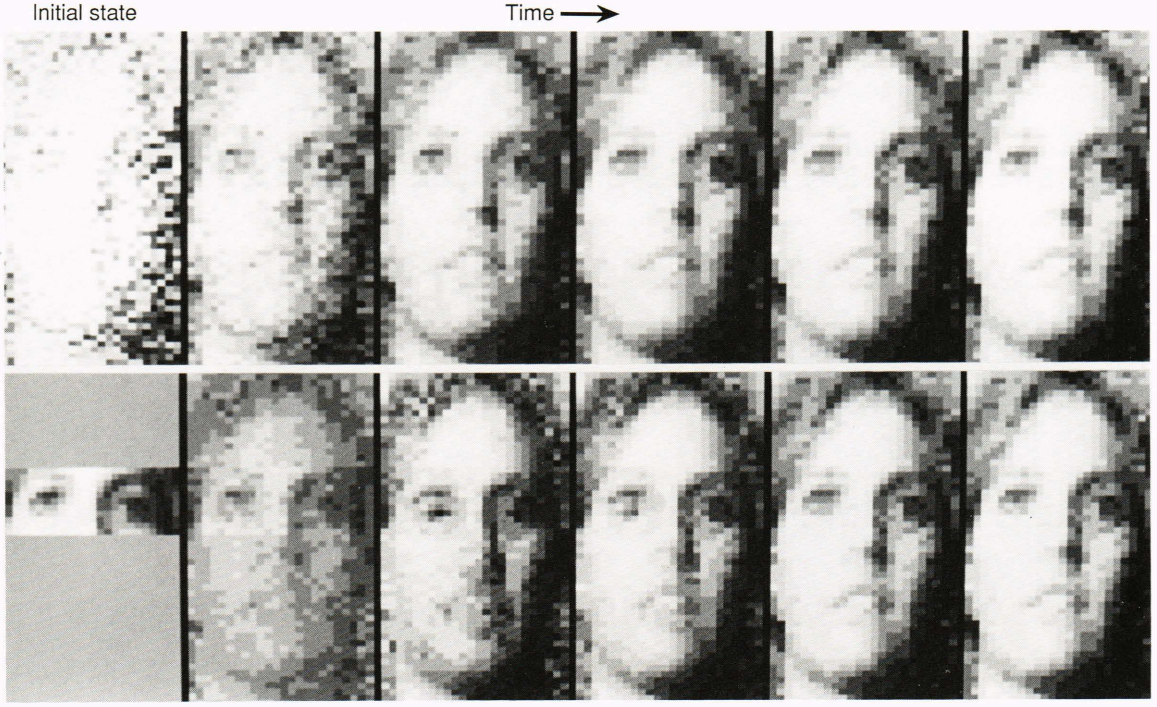


Figure 5—Response of the system of Fig. 4 after having learned six faces. The pictures are a grey-level encoding of the state of the units in the bottom layer. The sequence shows the time evolution of the network starting from the initial state on the left.

up to about 10% of the total picture size. Image or signal registration is a chronic problem in most real-world processing tasks, and the ability to deal with even small distortions of that type was encouraging. Other positive tests included destroying a number of connections in the network after learning, to demonstrate fault tolerance, and using networks with similar topology but which were sparsely connected. A negative result was the inability to store more than about a half-dozen faces in a network of 60,000 connections. The work is continuing with an emphasis on including second-order terms in the activity potential U_i of Eq. 1. In the second-order system, the potential is given by:

$$U_i = \sum_j \sum_k W_{ijk} X_j X_k \ .$$

There is evidence that both memory capacity and learning rates are significantly increased by introducing higher-order coupling terms.

Other numerical experiments with this software system have been performed by Ben Yuhas to support his Ph.D. research in the Department of Electrical and Computer Engineering at the Homewood Campus of The Johns Hopkins University. Yuhas's research is directed to an understanding of the relation between an acoustic speech signal and the corresponding visual signal of the speaker's lips. Such an understanding could supply techniques for automated lipreading or for improved speech recognition in noisy environments. As part of the effort, Yuhas is investigating the possibility of directly mapping digitized video images of a speaker's lips to the short-term power spectral density of the accompanying speech. He has performed a number of experiments using the pattern-mapping mode to successfully train networks to perform the mapping. Figure 6 is an example of the data he is working with, which have up to now been limited to static images of vowel and dipthong sounds.

One standard technique for such a mapping is to use the training data as a library. When a new lip image is encountered, the library is searched via some matching criterion, such as correlation, and the output spectrum is chosen from the closest matches. In another variation, the library can be used as a set of basis vectors to form a linear mapping. Yuhas has been able to show, for the limited cases examined thus far, that a network can reproduce the acoustic power spectrum somewhat better than the standard library techniques. This implies that some of the subtle variations in the acoustic signal are reflected in the lip image. Current plans are to expand the experiments by including information about the lip dynamics in the form of temporal image data, rather than just static images.

HARDWARE ACTIVITIES

Along with our numerical and theoretical activities, we have been investigating potential hardware implementations of the neurodynamic computing algorithms. There are two possible paths to be taken, digital or analog, and both are being examined. We feel that there is much near-term potential in connectionist algorithms as signal-processing tools, through a straightforward simulation of the network dynamics on conventional digital machines. Even greater potential can be realized through special-purpose digital processors, where the architecture has been tuned to the network simulation. The bigger, far-term potential lies in the analog approach.

On the digital side, there is already a good deal of commercial activity to develop products based on conventional pipelined array processors with software shells that allow easy implementation of a variety of neural nets. Our activities should not overlap that type of product development. The near-term products that will emerge from the commercial developers will likely strike a balance among generality, performance, and ease of use. Generality is a primary ingredient for a viable commercial product because of the need for wide appeal to generate larger-volume sales. That is not necessarily so for application in certain research and military systems.

If one is willing to sacrifice generality, the simulation of the dynamics is a good candidate for an application-

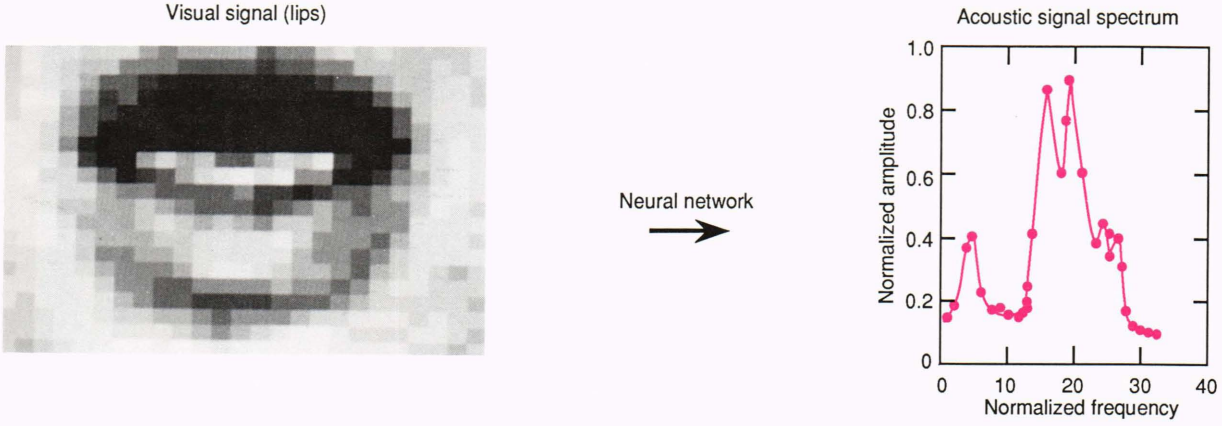


Figure 6—Examples of the visual-to-acoustic pattern-mapping data being used by Ben Yuhas to train a network to perform the mapping. Digitized video images of a speaker's lips are used as the inputs to the bottom layer of the network. Supervised learning is then used to force the outputs of the top layer of units to be the normalized short-term power spectral densities that correspond to those video frames. Those spectra can then be used to help reproduce the speech.

specific coprocessor of the extremely low-cost variety espoused in Ref. 10. That is, with a combination of custom chips and standard commercial parts in an architecture specifically tuned to Eqs. 1 through 6, it may be possible to get very high performance for very low cost. The major computation required is a sparse matrix multiply, which should pipeline quite well. In addition, preliminary studies have indicated that precision of only about 7 or 8 bits is needed in **W**. We have been examining architectural possibilities for a single-board personal-computer coprocessor for Eq. 1 that would perform at least 30 $\times$ 10^6 multiply/accumulations per second for weight matrices of several million connections. The target component and fabrication cost for the board is less than $1000.

Craig Vandervest, of the APL Fleet Systems Department, has developed a candidate architecture for such a board as a course project in his master's degree program. His approach is based on pipelining, and it capitalizes on the sparse-matrix representation and 8-bit precision requirement. Although his work has only recently been completed and is not yet documented, it has served as a starting point for further development of the application-specific approach. In a joint effort with the Computer Engineering Group of the Technical Services Department, that work will be continuing for the remainder of 1988. The work will include further investigation of the important issue of precision requirements.

Kim Strohbehn is leading the development of a completely different type of application-specific processor based on a connectionist architecture. It is a wafer-scale very-large-scale-integration (VLSI) implementation of an optimal detector for drifting, narrowband signals. The processor would operate on images formed by taking sequences of power spectra of a received low-level broadband signal. If the image rows are the sequential power spectra, then the image is known as a time–frequency image. A drifting, narrowband signal would appear in the image as a faint, hard-to-detect diagonal line. Detection of such a signal is a problem of general applicability, including the search for extraterrestrial intelligence.

Our approach is to use neural units that are each connected to an individual subfield of the time–frequency image (Fig. 7). The receptive field of each unit corresponds to a unique linearly drifting signal, and by theshholding the summed signal in that field the unit acts as a near-optimal feature detector for that line. We call the receiver a "brute force detector" (for obvious reasons); the idea for it arose from the work of Michael Roth of the APL Fleet Systems Department (see the article by Roth elsewhere in this issue). The outputs of the receptor units themselves form an image in frequency-slope space, and we envision a subsequent layer of units to detect patterns in the output.

A test chip for the receiver has been designed and fabricated in complementary metal oxide silicon (CMOS). The test chip contains two receptive units for prototyping. Laser reconfigurable test cells have been included on the same chip to explore the possibility of wafer-scale integration using techniques described in Ref. 11. The

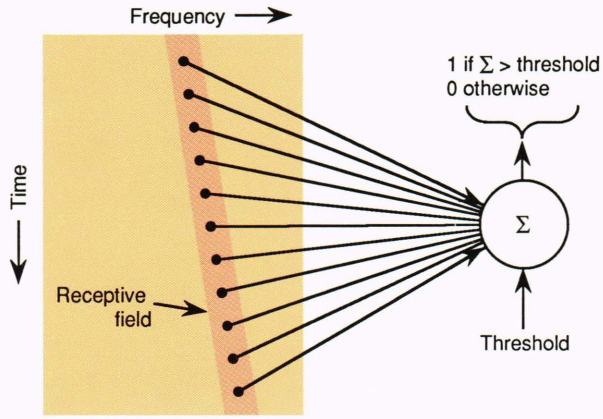


Figure 7—A neural-network detector for drifting, narrowband signals. Each unit in the second layer is a feature detector that accumulates pixel values in a specific region of a time–frequency image. That region is called the unit's receptive field. The unit effectively performs a noncoherent integration of a particular frequency and drift rate, acting as a near-optimal receiving element.

idea is to fill the wafer with neural units that can be assigned receptive fields, and then eliminate the nonworking elements by laser reconfiguration. Our results indicate that on the order of 10^5 receptive fields could be implemented on a 6-in. wafer using Strohbehn's design.

In collaboration with the Department of Electrical and Computer Engineering at the Homewood Campus, an initiative has also been started in the technology needed for analog implementations. This activity is, by design, lagging our theoretical and applications effort. Our approach is modeled after that of Professor Carver Mead at the California Institute of Technology. Dr. Mead is an innovative thinker who has had major influence in this country on the technologies associated with integrated circuits. He has recently mapped out an approach to analog computation that capitalizes on the high integration densities and low fabrication cost achievable with metal-oxide–silicon field-effect transistor (FET) devices. (In my opinion, it is possible to have a useful and productive research program in advanced computing through the simple strategy of trying to keep up with Carver Mead.)

In digital circuits, FETs are operated as switches controlled by a voltage applied to their gates. To turn on the device, the gate voltage is above the threshold voltage (the critical voltage for which a highly conducting channel forms between source and drain). When the gate voltage is below the threshold voltage, the switch is off. In the off state, the device operates in the subthreshold regime, with only very tiny (nanoamperes) diffusion currents flowing between source and drain. In digital applications such a subthreshold current is considered a parasitic nuisance. Mead, however, has emphasized that it has a repeatable logrithmic dependence on low-level gate voltage that gives it very desirable properties for analog computation. In a textbook soon to be released,[12] he has outlined the theory for subthreshold FET operation, described a number of circuits that can be used as computational elements, and illuminated a path toward ap-

plication of the technology to neural-network computations. Dr. Mead graciously supplied us with a preprint of that text to help us implement an analog VLSI course at The Johns Hopkins University.

Using the subthreshold techniques described by Mead, it should be possible to design application-specific VLSI chips with a good deal of analog computing power. Further, no special fabrication techniques would be required to achieve the normally high integration levels associated with metal-oxide-silicon chip fabrication. There is a tremendous advantage in cost and parts accessibility if such chips can be fabricated by the standard 1.5-μm CMOS foundry processes. The option of combining digital and analog circuitry on the same chip for signal-processing applications becomes viable. Also, the self-compensation techniques discussed by Mead may make analog processing inherently more radiation-resistant than digital processing in spacecraft applications. Thus, there are applications other than neural networks that could profit from the technology.

Through a joint effort with Dr. Andreas Andreou, a Research Associate in the Department of Electrical and Computer Engineering on the Homewood Campus, much progress has been made in establishing an analog VLSI capability there. During the past two years he has set up a chip-probe facility that can reliably measure currents as small as 1 pA, a crucial requirement for testing subthreshold analog chips. An analog-circuit-design laboratory course based on the preprint of Mead's textbook has been introduced at the Homewood Campus, and several student projects have been started. Those projects are designed to help us all become familiar with design and test techniques, to gain some experience with foundry-fabricated devices, and to start building a library of proven computational cells. One of the projects, a Bidirectional Associative Memory (BAM), has been fabricated in a prototype CMOS chip.

The BAM design was carried out by two students at the Homewood campus, Kwabena "Buster" Boahen and Philippe Pouliquen, and is based on a neural-network architecture discussed by Bart Kosko.[13] In that system, the units are arranged in two layers with two-way synaptic connections between the layers (Fig. 8). The units have two possible states (+1 and −1) that are determined by thresholding the sum of their weighted inputs. Kosko analyzed the autoassociative properties of the system and showed that it would store and retrieve binary patterns from noisy or incomplete initial conditions. The system is similar to the one being used for the face-recognition experiments (Fig. 4), but the use of two-way connections makes it effectively have a symmetric weight matrix so that its character is similar to that of a Hopfield net.

Boahen and Pouliquen have cleverly implemented the system using subthreshold analog techniques, and a tiny prototype chip has been fabricated in 3-μm CMOS. The chip (Fig. 9) contains 32 units and can be programmed by external digital input to store 8 binary patterns of 32 bits. The initial conditions for pattern retrieval are input into the chip using the same input pads as those used for programming the connection strengths. The fabricated chips have been tested, and a description of the work is being submitted for publication. Our plan is to expand the design to a full-sized (9-mm) chip in standard 1.5-μm technology, which will easily contain 256 units and be programmed for 256 stored patterns.

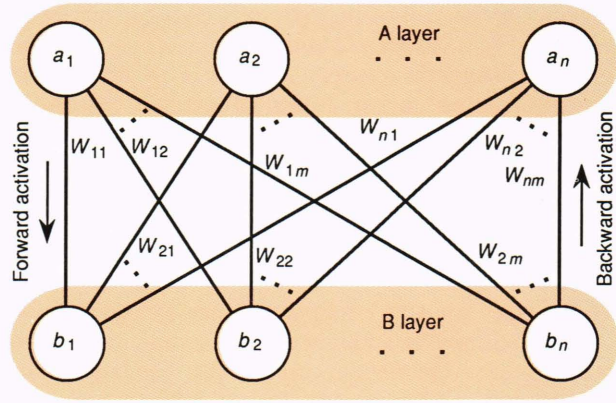


Figure 8—Architecture for a bidirectional associate memory. a_i and b_i are neural units with possible states of +1 and −1. The W_{ij} are synaptic connections that control activation in both directions. Memories are stored in the system as binary-pattern pairs.

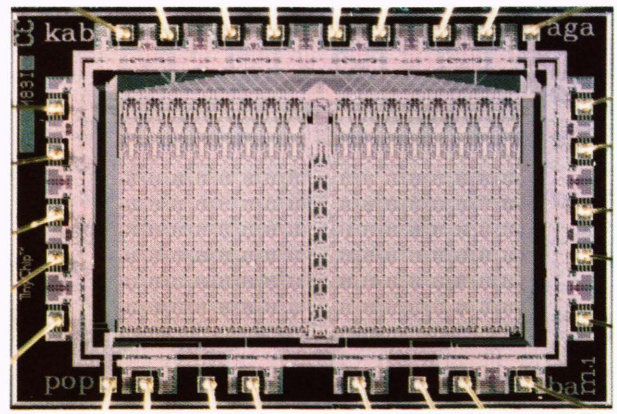


Figure 9—Photomicrograph of the BAM chip. The dense central area is the storage cells for the connection weights, along with the analog cells for forming the activity potentials from the weights. The periphery contains the interface to the outside world, including logic for externally loading the weights.

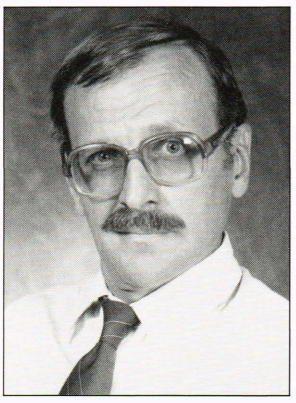
REFERENCES

[1] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing,* Vols. 1 and 2, MIT Press, Cambridge, Mass. (1986).
[2] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Natl. Acad. Sci. USA* **79**, 2554–2558 (1982).
[3] F. J. Pineda, "Generalization of Back Propagation to Recurrent Neural Networks," *Phys. Rev. Lett.* **18**, 2229–2232 (1987).
[4] F. J. Pineda, "Dynamics and Architecture in Neural Networks," *J. Complexity* **4** (special issue on neural computation), 181–211 (1988).
[5] G. A. Carpenter and S. Grossberg, "The Art of Adaptive Pattern Recognition by a Self Organizing Neural Network," *IEEE Computer* **21**, 77–88 (1988).
[6] A. Atiya, "Learning on a General Network," in *Proc. IEEE Conf. on Neural Information Processing Systems,* D. Z. Anderson, ed., Denver, Colo. (1987).

[7]G. Tesauro and T. J. Sejnowski, "A Parallel Network That Learns to Play Backgammon," submitted to *Artificial Intelligence.*
[8]J. J. Hopfield, "Neural Computation of Decisions in Optimization Problems," *Biol. Cybern.* **52**, 141–152 (1985).
[9]J. S. Denker, ed., *Neural Networks for Computing, AIP Conf. Proc. 151,* Snowbird, Utah, American Institute of Physics, New York (1986).
[10]R. E. Jenkins and D. G. Lee, "An Application-Specific Coprocessor for High Speed Cellular Logic Operations," *IEEE Micro* **7**, 63–70 (Dec 1987).
[11]J. I. Raffel, J. R. Mann, R. Berger, A. M. Soares, and S. Gilbert, "A Generic Architecture for Wafer-Scale Neuromorphic Systems," in *Proc. IEEE First Int. Conf. on Neural Networks,* San Diego, Calif. (1987).
[12]C. A. Mead, *Analog VLSI and Neural Systems*, preliminary notes for publication (1986).
[13]B. Kosko, "Bidirectional Associative Memories," *IEEE Trans. Syst. Man. Cybern.* **18**, 49–60 (1988).

ACKNOWLEDGMENT—Clearly this article describes a good deal of published work in the open literature, as well as the collective activities of a number of people within our group of collaborators—at APL: F. Pineda, K. Strohben, S. Yionoulis, G. Lee, J. Hayes, D. Redish, and C. Vandervest; and at the Homewood Campus: A. Andreou, B. Yuhas, K. Boahen, P. Pouliquen, and A. Pavasovic. As a group we have profited by the occasional advice and wisdom of T. Sejnowski at the Homewood Campus, one of the leading researchers in the field.

THE AUTHOR



ROBERT E. JENKINS was born in Baltimore and received an M.S. degree in physics from the University of Maryland in 1965. He joined APL in 1961 and is supervisor of the Computer Science and Technology Group, an elected member of the APL Advisory Board, a member of APL's standing Independent Research and Development Committee, the program manager for Space Department Independent Research and Development, a member of the Electrical Engineering Program Committee for The Johns Hopkins G. W. C. Whiting School of Engineering, and a lecturer in electrical engieering at both the Homewood Campus and the APL education center. During 1978, Mr. Jenkins was visiting scientist at the Defense Mapping Agency. In 1985, he was awarded the Dunning professorship at the Homewood Campus, where he introduced a new course in very-large-scale integration design and conducted research in cellular automaton processing.