MICHAEL W. ROTH

NEURAL-NETWORK TECHNOLOGY AND ITS APPLICATIONS

Neural-network technology can provide solutions to a wide variety of science and engineering problems that involve extracting useful information from complex or uncertain data. Although neural networks have long been a topic of academic research, the recent developments of effective learning algorithms and special-purpose computational hardware and the demonstrations of their applications have suggested that neural networks can provide useful tools for solving practical problems. This paper surveys some of the highlights of developments in neural-network technology in the areas of algorithms, hardware, and applications. Some of the ongoing work on neural networks at APL is also described.

INTRODUCTION

There are a number of problems in science and engineering that involve the extraction of useful information from complex or uncertain data. For many of these problems, the traditional approaches using techniques of signal processing, pattern recognition, control-systems theory, and artificial intelligence have been inadequate. Pattern recognition of fixed patterns in stationary backgrounds is a straightforward task for which numerous effective techniques have been developed. If the patterns or the backgrounds are variable in either a limited or known manner, more complex techniques—such as those using methods based on artificial intelligence—can be effective. But if the patterns or the backgrounds vary in an unlimited or unknown manner, the traditional approaches have not been able to furnish solutions.

The success rate of solving pattern-recognition problems using traditional pattern-recognition approaches has mirrored the success rate of artificial intelligence because much of the current pattern-recognition technology heavily uses the methods of artificial intelligence. Experience has shown that artificial intelligence is good at solving problems that can be strictly defined by rules. For many applications, however, such rules are difficult or impossible to develop. In general, artificial intelligence has problems with choice because of the frequent need to conduct a time-consuming search through large databases consisting of predetermined rules. Unfortunately, that approach has not been successful if the rules could not be firmly established, because of either complexity or uncertainty. A new approach is required—one that offers the possibility that it will do things better than artificial intelligence when there is high complexity or high uncertainty with regard to the definition of the rule set.

Artificial intelligence has been successfully applied to problems such as proving mathematical theorems. With the application of a large amount of effort, it has led to effective systems for chemical analysis and playing chess. But it has difficulty with seemingly simple problems, which in human experience employ common sense—such as that of understanding a simple world made from children's playing blocks. The difficulty stems from the very large number of rules required for employing common sense and the inability of current artificial-intelligence technology to deal with problems having a large number of rules. Artificial intelligence has yet to provide an accurate system for continuous speech recognition and has made little progress on image interpretation or sonar recognition.

Expert systems are merely those artificial-intelligence systems for which an appropriate known or knowable rule base can be developed by an analyst. While expert systems have been effective in applications where a real expert could contribute to the definition of a real effective rule set, expert systems have been ineffective when there was no real expertise or when there was an inability to define an appropriate rule set, such as in attempts to create expert systems for stock-market trading. Expert systems based on artificial-intelligence techniques exhibit brittle rather than robust behavior; that is, there is great sensitivity to the specific assumptions and environments.

Likewise, numerous other science and engineering problems that depend on effective pattern recognition have had limited or little success when the pattern-recognition rules became either too complex or too uncertain. Similar remarks can be applied to related kinds of problems for signal-processing and control systems. Such problems have included: pattern classification; real-time artificial intelligence; data fusion; image-data restoration, compression, and recognition; scene analysis; control systems under anomalous sensor/servo behavior; knowledge extraction from large databases; and autonomous vehicles. The solutions to those and similar problems require the application of a new technology with the potential for improved performance when there is uncertainty or high complexity with regard to rules.

Neural networks are a developing technology that uses the massively parallel-distributed processing potentials of computational hardware that can only now be realized.[1-6] Neural-network technology is aimed at developing information processing that is analogous to biological nervous systems. Neural networks are the models and algorithms that can be simulated on conventional computers but are best implemented on special-purpose computational hardware. Known as neurocomputers, they exhibit characteristics that are not readily available in other types of systems. These characteristics suggest the potential for new systems capable of learning, autonomously improving their own performance, adapting automatically to changing environments, and coping with serious disruptions.

Neural networks and neurocomputers represent a radical departure from digital-computer algorithms and architectures. They are "neuron-inspired," using processing elements that share some of the properties of biological nervous systems. Neural networks do not attempt to simulate accurately real neurons. They are attempts to implement approximations of useful computational properties exhibited by biological nervous systems. The technology underlying the development of neural networks draws heavily from: (1) cognitive psychology models of human memory, learning, and perception, (2) biological models of synaptic organization of neurons, and (3) the device physics of special-purpose computational hardware.

Although conventional computer technology has progressed to an advanced, powerful, and affordable state, biological nervous systems can solve problems and perform calculations that are well beyond the abilities of supercomputers. Unlike conventional computers, neural networks seek to emulate aspects of biological systems, and they have shown that they can spontaneously learn or can discover new ways to process input data and modify the rules they use so as to seek the best solution to a given problem; conventional computers are programmed with hard-and-fast rules that must be manually reprogrammed if a better solution is needed. Neural networks can process inexact, ambiguous, fuzzy data that do not exactly match any information stored in memory; conventional computers cannot adjust the specific definitions and rules that they are programmed to use so as to accommodate new, inexact, degraded, or contradictory input. Neural networks are very good at solving knowledge-processing problems by virtue of their ability to do hypothesis testing, to detect statistical patterns and regularities, to perform probability estimation, and to adjust dynamically the implicit rules used to process information when presented with new input data. Neural networks can offer better solutions to some information-processing problems currently being addressed by the expert-systems sector of traditional artificial intelligence. Although some concern has been expressed as to whether neural-network solutions could be validated only empirically, researchers are developing tools so that both specific network elements as well as networks as a whole can be validated. (See the boxed insert for some abilities that have been demonstrated in simple neural-network systems).

In the remainder of this paper, some of the highlights of neural-network technology are surveyed. Unfortunately, because of the very large number of publications concerning neural networks, no single paper could even hope to survey the entire field. The application of neural networks to a broad area of problems is a relatively recent development, and this paper represents an attempt to survey some of those applications and related models, algorithms, and special-purpose computational hardware. For surveys and reviews of other aspects of neural networks, see Refs. 1 to 6.

SOME ABILITIES DEMONSTRATED IN SIMPLE NEURAL-NETWORK SYSTEMS

Real-time performance in pattern recognition
The ability to recognize patterns with computational speeds at rates at least as fast as the rates at which new patterns can be entered into the system

Control in multiple-constraint environments
The ability to control systems in environments with a multiplicity of complex nonlinear constraints

Associative recall on cue
The ability to retrieve original inputs from fragments of the original

Robust associative recall
The ability to recognize distorted patterns

Intelligent association
The ability to "remember" related items even when the relationship might not be obvious or carefully predetermined

Real-time learning
The ability to learn (in real time) solutions to new problems, from positive demonstrations of solutions to similar problems, with the ability to adapt to changing environments

Graceful degradation
The ability to recall memories even if some individual processors (units) fail—stored information is distributed among many processors and their interconnections, so that performance degrades gracefully rather than catastrophically in the event of unit failure

NEURAL-NETWORK MODELS AND ALGORITHMS

The basic unit of a neural network is the processing element that performs the operation

$$y_i = f\left(\sum_j W_{ij}\, x_j - T_i \right) , \qquad (1)$$

where x_j are the input values; y_i are the output values; W_{ij} are the interconnection weights; T_i are the threshold values; and f is the response function that equals 1 for large positive arguments, 0 for large negative arguments, and varies monotonically between 0 and 1 for intermediate values. The processing element given by Eq. 1

is the starting point for most neural-network models, algorithms, hardware, and applications. When the response function f in Eq. 1 is a step function, the processing element is referred to as a threshold-logic unit. A neural network consists of a network of processing elements in which the units may connect onto either themselves or additional units. Additional terms, such as dissipation terms, may also be added to give the network dynamic behavior. The importance of dynamic behavior is described later in this paper. The motivation for creating such networks of units is that when a sufficiently large number of units (with an appropriate selection of weights) are connected to form a network, significant computations can result. The selection of the values of the weights W_{ij} is one of the most important aspects of neural-network technology because it determines the specific computation that the network performs, such as pattern recognition. The specific values of the weights are determined by algorithms referred to as learning algorithms.

The Hopfield Model and Its Extensions

Much of the current interest in neural networks was stimulated by the work of Hopfield and his collaborators.[1,7-15] In 1982, Hopfield introduced a model in which the processing element was a threshold logic unit, the weights were selected as the sum of the outer products of desired pattern values as suggested by Hebb,[16] all the units were connected to each other, and the units were updated in a random, asynchronous, and recursive manner.[7] The motivation for this model was to show that a network of simple, nonlinear, recursive units could achieve significant computational capabilities. Because of the symmetry of the weight matrix, Hopfield showed that this model could be interpreted in terms of an energy function such that stable states of the model corresponded to minima of the energy function.

Figure 1 shows how the Hopfield model acts as a content-addressable memory, with the desired memories representing minima of the energy function. The computational energy of the Hopfield model can be pictured as a landscape of hills and valley. The connection pattern and other characteristics of the model determine its contours. The model computes by following a path that decreases the computational energy until the path reaches the bottom of a valley, just as water moves downhill to minimize its gravitational potential energy. If the model starts out with approximate or incomplete information, it follows a path downhill to the nearest valley that contains the complete information.

Although the Hopfield-82 model was an important demonstration of the significant computational capabilities that a neural network of simple units could achieve, it had a number of limitations that motivated additional efforts. One of its main limitations was that the storage capacity of the network (in terms of the number of patterns that could be stored with little or no error) was about 14% of the total number of units. Several studies have been conducted to quantify precisely that storage limitation.[17-21] Another limitation was the existence of the so-called "spurious" states—stable final states that were not members of the set of originally selected patterns. Because of group theoretical properties of the Hopfield-82 model, those spurious states can be identified precisely.[22]

In 1984, Hopfield introduced a significant extension[9] of the Hopfield-82 model to overcome the computationally intensive procedure of asynchronous updating that was necessary in the original model for the system to converge to global rather than local minima.[23] Unlike the earlier model, the Hopfield-84 model was synchronous, continuous, and deterministic with a sigmoidal response function $f(u)$. Hopfield showed that there is also a corresponding energy function such that in the high-gain limit, the system is equivalent to the original Hopfield-82 model. The Hopfield-84 model inspired many efforts at analog VLSI (Very Large Scale Integrated) circuit implementations and computational experiments because of the advantages of the deterministic formulation over the stochastic one. For example, the Hopfield-84 model retains the ability of a content-addressable memory similar to the 1982 model.[24] The sigmoidal nonlinearity is an important feature of the model because a sharp nonlinearity causes convergence to local rather than global minima.[10,12] The importance of the sigmoidal nonlinearity has also been emphasized by Grossberg.[6] Convergence theorems for continuous models with sigmoidal nonlinearities have also been proven.[25]

The most straightforward technique for increasing the storage capacity of the Hopfield model is to orthogonalize the desired patterns. This is a well-known technique for linear neural networks and can be implemented by using a weight matrix related to the Moore–Penrose pseudoinverse.[26] Orthogonalization and use of the pseudoinverse greatly increase the storage capacity of the Hopfield models.[27-30] It is even possible to have perfect storage and to eliminate spurious states altogether by storing not only orthogonal patterns, but also faithful patterns that are exactly and exclusively reproduced by the model.[31]

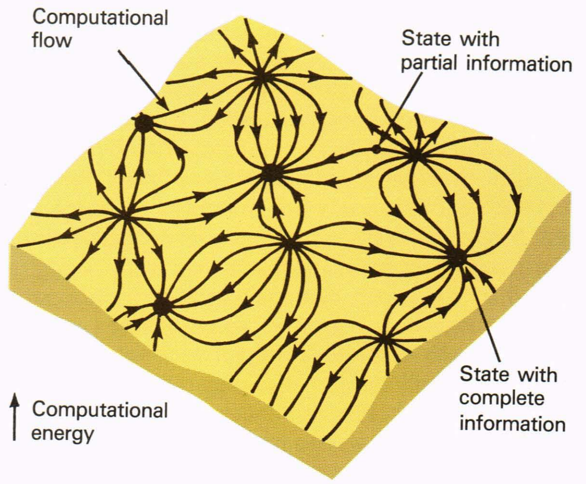

Computational flow
State with partial information
State with complete information
Computational energy

Figure 1—A representation of the computational flow that minimizes the computational energy for the Hopfield neural-network model (adapted from Ref. 15).

Higher-order extensions of the Hopfield model are also of interest because of their relationships with the higher-order energy functions for certain optimization problems.[32-34] In those networks, the input to the response function is given by the sum of the product of a weight tensor and unit values, so that Eq. 1 is replaced by

$$y_i = f\left(\sum_{j_1 \cdots j_n} W_{ij_1 \cdots j_n} \, x_{j_1} \cdots x_{j_n} \; - \; T_i \right) , \qquad (2)$$

where the order of the weight tensor is $n + 1$. Numerical simulations show that use of these higher-order interactions gives better memory capacity than the Hopfield-82 model.[35,36] The pattern-storage capacity increases by the number of units to the power of n, but it is completely compensated for by an increase in memory-storage requirements for the tensor weight matrix.[37] The information capacity (in terms of the ratio of the number of bits stored to the number of bits required for recall with a fixed error rate) is also independent of the order.[38] A single layer, as in Eq. 2, however, can have translation invariance, as well as other geometric invariances and *a priori* knowledge, automatically incorporated in an elegant fashion.[39]

Layered Architectures and the Backpropagation Algorithm

A different approach to neural networks that has also stimulated much current interest is the development of the backpropagation learning algorithm for layered, feed-forward networks. Although algorithms similar to backpropagation had been independently proposed by several different authors,[40-43] it was the studies of Rumelhart, Hinton, and Williams[44] that provided demonstrations and dissemination that did not previously exist. Backpropagation is a learning algorithm designed to solve the problem of choosing weight values for hidden units in a layered, feed-forward network. Hidden units are units in intermediate layers that are neither input nor output units. They are introduced to give the network enhanced internal-processing capabilities it could not have if only input and output units were present. The limitations of networks with only feed-forward input and output units, such as the single-layer perceptron,[45] were well documented by the work of Minsky and Papert.[46] The introduction of one hidden layer allows the network to represent an arbitrary Boolean function,[47] and two layers allows the network to represent an arbitrary decision space.[3,48,49] Hidden unit layers can be introduced to automatically represent geometrical invariances such as translation invariance.[50] Finally, the minimum number of layers and the number of units within hidden layers necessary to compute an arbitrary dichotomy of n points in general positions in arbitrary Euclidean dimensions can also be calculated.[51]

Although the introduction of hidden units gives a feed-forward network the potential for an arbitrary mapping, before the introduction of the backpropagation algorithm no known technique existed for determining the weights of a deterministic feed-forward network with hidden layers. This was the essence of the credit-assignment problem: how to determine the weights of intermediate hidden units such that desired responses occurred as a result of specified inputs. Several algorithms were known for single-layer feed-forward networks without hidden units, such as the perceptron learning algorithm[45] and the Widrow–Hoff algorithm.[52] In addition, an ingenious algorithm known as the Boltzmann learning algorithm[53] had been developed for a stochastic network, which used Bayesian probability rules to update the weights for both hidden and external units.

The backpropagation algorithm is an extension of the Widrow–Hoff algorithm and can be understood by the following. Define the error E between the actual outputs y_i and the desired outputs y_i^D as

$$E = \langle (y_i - y_i^D)^2 \rangle , \qquad (3)$$

where the angle brackets represent an average over all of the input patterns and output units. A gradient-descent technique for adjusting the weights so as to minimize the error leads to the expression

$$\begin{aligned} \Delta W_{ij} &= -\,\epsilon \, \frac{\partial E}{\partial W_{ij}} \\ &= -\,2\epsilon \left\langle (y_i - y_i^D) \, \frac{\partial y_i}{\partial W_{ij}} \right\rangle , \end{aligned} \qquad (4)$$

where ΔW_{ij} is the weight adjustment and ϵ is a selected adjustment rate. By choosing a conveniently differentiable response function f, and by applying the chain rule of calculus to compute the weight adjustments for layers before the output layer, a convenient formula for the weight adjustments of all layers can be derived, which has become known as the backpropagation algorithm.

The backpropagation algorithm has demonstrated several advantages in addition to having the potential for determining networks with arbitrary mapping properties. For example, backpropagation classifiers are reasonable alternatives to traditional classifiers for problems with continuous-valued inputs.[54] Although the error rate of a backpropagation-based pattern classifier often is close to that of a Bayesian classifier, a backpropagation classifier also is robust and can outperform conventional techniques by a wide margin when inputs have disjoint probability distributions. The backpropagation algorithm can also be used to learn distributed representations.[55] A distributed representation is one whereby a concept is distributed over several units, as opposed to being represented by a single unit. Using backpropagation, weights can automatically evolve to show relationships that were not explicitly input. Whereas neural networks can learn topological mappings using lateral inhibition and modified Hebbian-type learning,[26] the backpropagation algorithm can also be used to discover topological mappings[56] and to invert a feed-forward network.[57] Given a network of fixed weights and specified output patterns, the input patterns can be determined. Indeed, it has been proposed that backpropagation can be modified to minimize the log-likelihood function rather than the error, and thereby learn probability distributions.[58]

The gradient-descent nature of the backpropagation algorithm leads to slow convergence, and modifications to the algorithm have been proposed to speed up convergence. Adding a momentum term to Eq. 4 to smooth the weight-updating process has been shown to improve the convergence rate.[59] In addition, there are significantly different results when there is massive training of a particular item versus distributed training, which is better for a number of items.[60] This is to be expected from inspection of Eqs. 3 and 4 because the error is the total error over all the input patterns. Even the simple modification of shifting the response function to cover the interval $[-0.5, +0.5]$ results in much better convergence and much less variance for a large number of hidden units and patterns.[61] Sometimes the backpropagation algorithm gets stuck for a large number of iterations on an error plateau; it has been proposed that an order of magnitude improvement in convergence can be achieved by searching along the line of the error gradient and selecting the best point for weight updating.[62] Another suggestion is that a dramatic speedup can be accomplished by assigning a specific adjustment rate ϵ_p to each input pattern, by doubling ϵ_p for a particular pattern p if its individual error is too high, and by halving all the rates overall if the total error does not decrease very much.[63] Finally, extensions of backpropagation based on quasi-Newton methods of functional minimization have been introduced to speed up convergence, although such algorithms can have stability problems.[64-66] Unfortunately, because these algorithms have a higher order of complexity than backpropagation, they are appropriate for moderate problems but not large-size problems.

Despite the successes of the backpropagation algorithm, it has a number of limitations that affect its performance and potential for future applications. For example, the dependence of convergence on the number of hidden layers shows a complicated behavior.[59] The backpropagation algorithm sometimes fails to properly converge to bimodal patterns, and it has been suggested that randomization of weights, in addition to the initial randomization usually employed, may be fruitful in escaping such local minima.[49] Initial weight randomization is very important to the backpropagation algorithm. In fact, if a hidden layer of sufficient size with random weights is used, then even the perceptron learning algorithm can learn an arbitrary linear discriminant network.[67] Still, the weights most likely to be altered by the backpropagation algorithm are those that are already providing useful feature detection.[68] Some means needs to be identified to keep the useful features while exploring new possibilities. Backpropagation is good at generalization but poor at learning specific instances.[69]

One of the major limitations of backpropagation is that it does not scale well to larger-size networks. With more than a few thousand connections, it learns extremely slowly.[70] Perceptron-like learning algorithms require more than 2^d adjustments, where d is the number of input features[71] with similar results for backpropagation.[72,73] Part of the problem is that supervised learning in feed-forward networks is an *NP*-complete problem.[74] *NP*-complete problems are mathematical problems for which solutions require a number of computational steps that grow faster than any finite power of some appropriate measure of the size of the problem.

One of the most important limitations of backpropagation is its restriction to feed-forward networks only. As was discussed earlier, feed-forward networks can perform a number of computationally significant operations, but there are a significant number of computational operations that cannot be performed by a feed-forward network with a finite number of layers, as noted by Minsky and Papert.[46] An example of such an operation is determining whether a given figure is connected. But such a restriction does not apply to recursive or feedback networks such as the Hopfield model. As an illustration, Fig. 2 shows a three-layer network, with two of the layers recursive, that can compute the connectedness of a figure. The first layer is a lateral inhibition network that merely picks out a single point in the figure (noise can be added to ensure that only a single point is selected). The result is directly transferred by feed-forward connections to a Hopfield network whose weights are the outer product of the binary input pattern restricted to nearest-neighbor weights only. This second layer acts like a cellular automaton in that, if a single point of the input figure is activated, then all points connected to that point will also be activated. Finally, the result of the Hopfield network layer is fed forward to a single-threshold logic unit whose weights are the input pattern, but with a threshold sufficiently high so that a perfect match is required. The weights of the second and third layers are determined by only the current input pattern. In this manner, a network with a combination of feed-forward and recursive layers can compute the connectedness of a figure and overcome the computational limitations of feed-forward networks alone.

Some authors have begun to explore appropriate learning algorithms for such networks with both feed-forward and feedback connections. Lapedes and Farber[75] present an algorithm for implementing a content-addressable memory using a nonsymmetric neural network. Optimization of an energy function is shown to lead to dynamic equations that are related to the Hopfield-84 model. The learning equations are interpreted as another network analogous to the process of backpropagating the errors in the backpropagation algorithm. Unfortunately, the computations are very intensive because the number of weight equations is on the order of the square of the number of units. Pineda[76] makes a major extension of that work by first noting that feed-forward processing elements are a special case of the Hopfield-84 dynamic equations (e.g., the weight matrix

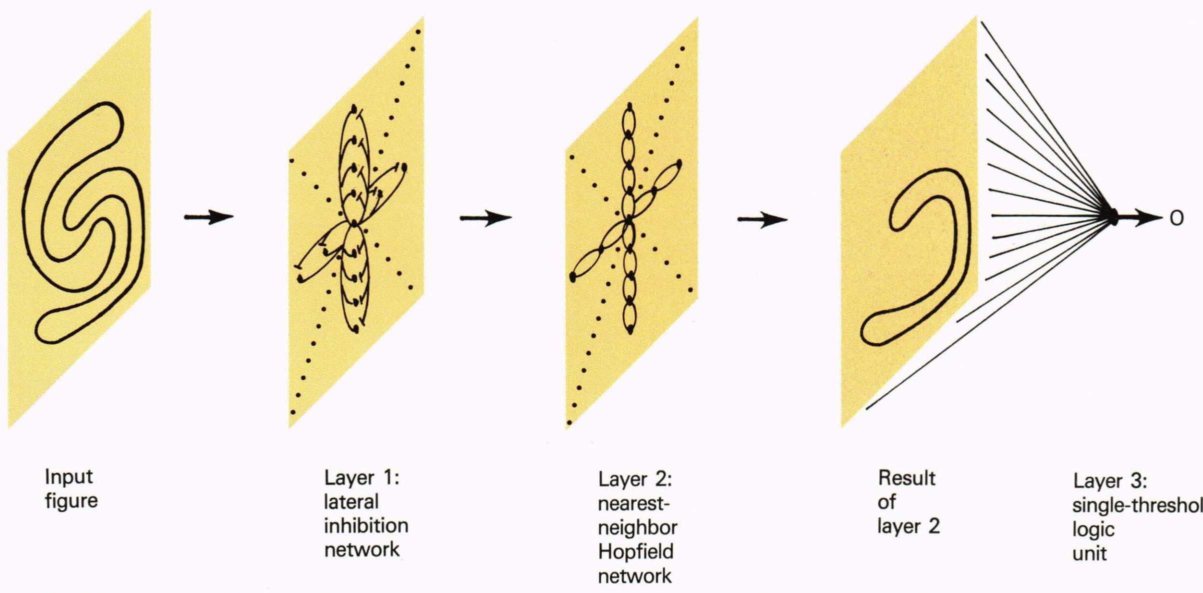

O
Input figure
Layer 1: lateral inhibition network
Layer 2: nearest-neighbor Hopfield network
Result of layer 2
Layer 3: single-threshold logic unit

Figure 2—A layered neural-network architecture for computing the connectedness of a given figure using both feed-forward and feedback connections.

is lower triangular) and then derives a learning algorithm for a network with arbitrary feed-forward and feedback connections for which backpropagation is the special case, limited to feed-forward networks only. (For a detailed derivation of the connection between Hopfield and feed-forward networks, see Ref. 77.) The learning equations can also be interpreted as another network, but with substantially improved computational performance over that for Lapedes and Farber. This algorithm cannot serve as an associative memory, because all the initial states are in a single basin of attraction.[78] The problem can be circumvented by deriving a related algorithm whereby the external units are constrained during learning to be the desired patterns. Almeida[79] independently derived a learning algorithm similar to Pineda's algorithm and showed that a feedback network with a smaller number of weights can at times achieve superior performance over a feed-forward network.

NEURAL-NETWORK HARDWARE

Neural-network models and algorithms are computationally intensive on general-purpose computers. Because of the computational simplicity of the basic processing element, however, neural networks can be implemented on special-purpose massively parallel hardware, which can vastly outperform implementations on even the most powerful serial computers. Consequently, a number of groups are developing such special-purpose neurocomputer hardware for implementing neural-network applications. The recent appearance of those neurocomputers has been an essential ingredient for the development of practical applications of neural-network technology.

The first generation of neurocomputers was based on pipelined implementations of digital VLSI technology with some low-level parallelism. In fact, a number of designs, such as the TRW, Inc., Mark III[80,81] and the Science Applications International Corporation (SAIC) Sigma-1, are now commercially available.[82] Those machines can handle on the order of 1 million connections, 60 thousand processing elements, and equivalent processing speeds that are 10 to 30 times that of a Digital Equipment Corporation VAX 11/780 computer. Neural networks have been implemented on more general-purpose parallel hardware, and an implementation on the Connection Machine 2 was reported to achieve an equivalent speed of 500 times that of a Digital Equipment Corporation VAX 11/780 computer.[83] Also, a number of special digital VLSI chips are being developed.[84] This first generation of neurocomputers consists of simulators of neural-network models and algorithms, and does not exploit the powerful computational potential of direct implementations using device physics.

A number of groups have recognized the computational potential of direct implementations of neural networks using the device physics of analog VLSI. Such efforts include the electronic neural-network chips developed at AT&T's Bell Labs,[85] a VLSI chip implementation of the Boltzmann learning algorithm,[86] and the analog VLSI retinal and cochiear chips developed by Mead.[87,88] The main advantage of going to analog VLSI is in using the analog circuitry to perform the neural-network computations, thereby gaining an enormous processing advantage. One Bell Labs chip has 256 processing elements and 130,000 fixed resistive weights, and can converge in less than 1.4 μs. That chip can be applied to a number of optimization and pattern-recognition problems that are described later. It is believed that this technology may be limited in the number of connections that can be achieved because of the two-dimensional nature of the chips.

Optical implementations of neural networks have the potential for achieving very high connectivity because beams of light can pass through one another without interaction.[89,90] There may be a fortuitous marriage between optical-computer technology and neural-network algorithms because of their complementary strengths and limitations. Future applications of neural networks will require massive parallelism, which is a strength of optical computers. Optical computers have limited dynamic range, which may not be a problem for neural networks. Finally, there are possibilities for all-optical computation loops that would avoid the interface bottleneck between electronic and optical components. Optical neuro-computers are prime examples of the computational potential for neural-network models and algorithms to exploit device physics.

The first optical neurocomputer designs employed optical matrix–vector multipliers using light-emitting diodes, photodiodes, light masks, and electronic feedback.[91,92] This matrix–vector–multiplier approach has evolved considerably to the point where designs have now been proposed with optoelectronic integration on VLSI chips. Photodiodes integrated into VLSI chips could provide optically controlled weights, and the light intensity could be varied by masks or acousto-optical crystals.[93] Another chip design proposes using magneto-optical spatial light modulators.[94]

An additional optical approach to implementing neural networks is the use of holographic devices such as volume holograms. Combined with optical feedback, systems could retrieve associatively millions of image patterns in a very short time.[89] The use of photorefractive crystals represents a promising approach to volume holograms because of the ease of dynamic holographic modification of interconnections.[95] The density of interconnections that may be implemented in these crystals is of the order of 10^8 to 10^{10} weights/cm^3. Also, the volume hologram can be viewed as a programmable two-port device that operates on the optical electric field.[96] The Hopfield model can be precisely implemented in an all-optical design using computer-generated holograms,[97] and a design for an all-optical implementation of back-propagation has been presented.[98] Holograms used in conjunction with optoelectronic resonator cavities have also been considered.[99] One of the most promising designs is the proposal by Caulfield to use page-orientated holographic memory.[100] In this design (Fig. 3) a number of small holograms are illuminated by a reference beam so that the holograms are imaged onto a transmissive or reflective spatial light modulator that is used as a representation of the input pattern. The resultant image—the sum of the holograms weighted by the input—is extracted to form the output.

APPLICATIONS OF NEURAL-NETWORK TECHNOLOGY

One of the first applications of neural-network technology was for the solution of complex optimization problems. Because many different kinds of problems can be formulated as optimization problems, the potential for neural-network technology to solve quickly such problems is a very important development. The first demonstration of such a potential was by Hopfield and Tank[10] for the Traveling Salesman Problem. (The Traveling Salesman Problem is an *NP*-complete problem that asks what is the shortest tour of several cities such that each city is visited only once.) Hopfield and Tank showed that the Hopfield-84 model can be used to compute good solutions to that problem within a few network time constants. They also showed that the Hopfield-84 model can be configured to solve decomposition problems and related linear-programming problems.[11] Other kinds of optimization and constraint satisfaction problems, such as constructing a four-color map[102] and detecting graph isomorphisms,[103] can be solved by the same techniques. Another application is for communication modems, where the maximum-likeli-

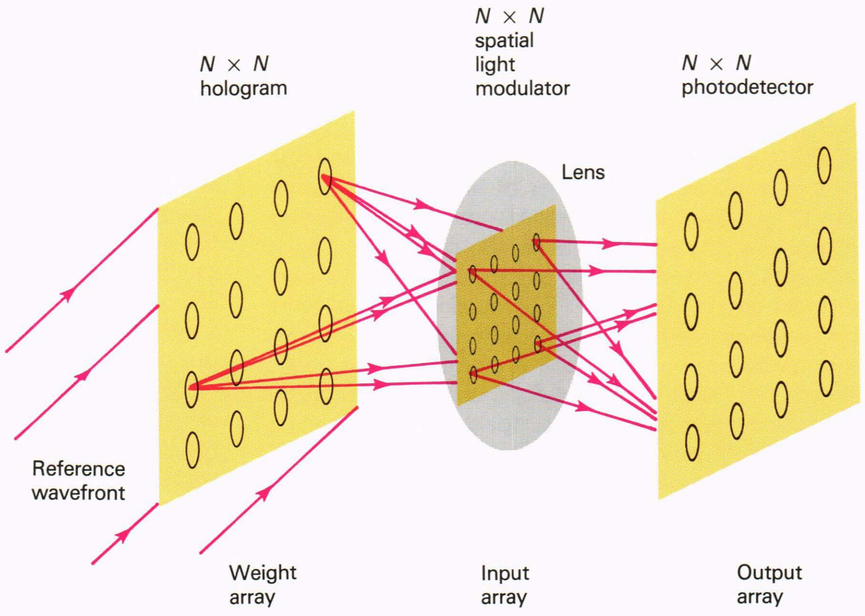


Figure 3—An optical neurocomputer design using page-orientated holographic memory, with the potential for 10^{12} interconnections. Data input is achieved with a spatial light modulator array. Each element of the input array is combined with a weighted array from a holographic array into an output array (adapted from Ref. 101).

hood-sequence estimator can be implemented by the Hopfield-84 network with much less complexity than the Viterbi algorithm.[104] Both static and dynamic scheduling of multiple microprocessors are also optimization problems that can be solved by those techniques.[32,33] Finally, the problem of computing image shape from image shading can be interpreted as an optimization problem that can also be solved by a modified Hopfield-84 model.[34]

The Hopfield model can also be used to perform pattern recognition. Farhat et al.[105] apply a hetero-associative variant of the Hopfield model to the classification of radar targets. Because radar returns consist of dilute, point-like images, the data are represented as sinograms that consist of the variations of the locations in an image of the dilute points as a function of the aspect angle. This results in good classification with as little as 10% of the sinogram available. Neural-network implementations related to the Hopfield-84 model can also be used for extraction of weak targets from high clutter environments[77] and multitarget tracking.[106] Application of the Hopfield model, using the pseudoinverse weight matrix, to handwritten character recognition gives results comparable to computing Hamming distances or optimal linear classifiers.[30] This pattern-recognition application has also been used to design a system for video-data compression using a hierarchy of Bell Labs' electronic neural-network chips.[107] Finally, the Hopfield-84 model can be extended for recognizing time-warped and noisy sequences with application to continuous speech recognition.[13,14]

Because of backpropagation's ability to learn effective pattern recognition from training databases, it has found numerous applications in a variety of problems. The first such effective demonstration was by Sejnowski and Rosenberg,[108] who showed how a backpropagation network could learn to convert text to realistic speech. The backpropagation algorithm can also be used for recognition of spoken digits[48] and handwritten characters[48,109] with excellent performance. Additional applications include classification of electronic intelligence data[110] and recognition of sonar[111] and radar[112] targets. The pattern-recognition capacity of backpropagation can also be used for data compression by forcing the output to reproduce the input for a network with a lower-dimensional hidden layer.[113] Such a technique has been applied to aircraft infrared data, after Fourier polar transformation to remove translations and rotations, and showed that the hidden layer did separate the data into clustered classes.[114]

Neural networks have been applied for a number of years to solve engineering problems in the area of control systems.[115] One example is the use of polynomial networks to model the optimum solution for a large domain of possible missile-flight trajectories in the two-point boundary-value problem. More recently, backpropagation has been used to train a neural network to drive an automobile.[116] To avoid control-system instability, the network first attempts to imitate a trainer, then it is permitted to operate with the trainer overriding mistakes and retaining proper performance. Several authors have proposed neural-network models for explaining coordination of head and eye movements in animals, and the weights of the network have been determined using both the pseudoinverse[117] and backpropagation.[118] Those models can also be used for robotic systems. Finally, because optimal control systems are determined by minimization of a Liapanov function and the optimal trajectories could be approximated by training, optimal control is a promising area for future applications of the Hopfield model and the backpropagation algorithm.[66]

For applications related to expert systems, neural networks can implement propositions and constraints with the ability to backtrack for explanations.[119] They also have features that support both robust reasoning and knowledge acquisition. Inferencing techniques can be developed to allow the expert system to reach conclusions when only a fraction of the input values is known.[120] Also, techniques for confidence estimation, question generation, and conclusion explanation can be developed. Expert systems using these techniques have been developed for medical diagnosis and other problems. The backpropagation algorithm has been applied to the development of expert systems as well.[121] As a demonstration of the potential for this approach, neural networks have learned to play backgammon by observing a human expert and training on the set of observations.[122] Finally, neural-network expert systems employing backpropagation have found a number of commercial applications for computer-aided decision systems for problems such as bank-loan advising and airline scheduling.[123]

There are a number of other applications of neural networks. One such application is for computational modeling of brain functions,[6,117,118,124] which is too extensive to describe here. Present and future neurocomputers have applications beyond the previously described neural-network models and algorithms because the hardware architecture consists of massive parallelism of simple processing units. Because any finite-state computing machine can be computed with a set of Boolean functions and because neural networks can be configured to compute an arbitrary Boolean function, any finite-state computing machine can be computed with neural networks. But some algorithms can be more efficiently implemented on conventional rather than neural-network hardware. Conversely, many well-known processing algorithms can be efficiently implemented using neural-network algorithms and hardware.[125]

ONGOING EFFORTS AT APL IN NEURAL-NETWORK TECHNOLOGY

APL's neural-network efforts involve the theoretical development of improved models and algorithms, the design of neurocomputer hardware, and the practical applications of existing models an algorithms. Roth[77] has shown how neural-network technology can be used to enhance the detectability of weak targets in high-clutter environments. In general, detection devices must set high thresholds to achieve a reasonable false alarm rate. This is especially true for environments, such as radar at low elevation angles, where the clutter distributions have long tails. The problem with setting a high threshold to cut down on false alarms is that detections of targets of small and medium size can be missed. A previously impractical idea for dealing with this problem was to implement a large bank of matched filters to cover the target variations over multiple scans. Because neural-network hardware is precisely designed to implement massively parallel computations, it offers new opportunities to implement such previously impractical ideas. In particular, it was shown that a modified Hopfield-84 model can implement the optimum post-detection target-track receiver. In this application, the desired states represent single target tracks, and spurious states would correspond to multiple tracks. Since it is desirable to detect multiple targets, the spurious states for this application are not a drawback but rather a desirable feature. Simulations have been conducted and show that considerable improvement of the signal-to-noise ratio can be achieved.

Figure 4a shows a simulated field that would correspond to multiple scans of a surveillance radar. The false alarm rate is chosen to be 2%. There are also two target tracks within the field with a probability of detection per cell of 10%. This field is input as the initial field of the modified Hopfield-84 model and is allowed to search iteratively for target tracks. Figure 4b shows the state of the network units after a few iterations. The results are false-color coded (black, blue, green, yellow, red, pink, white) for display purposes, with black and white representing 0 and 1, respectively. The target tracks are already visible as green lines. In following iterations, the tracks are much more visible while the clutter has been greatly reduced. This process continues until the clutter is eliminated and only the target tracks remain (Fig. 4c).

The previous simulations were performed on a serial computer and required considerable processing time. Special-purpose computer hardware is being developed, such as the digital VLSI chip designed by Strohbehn[126] for sonar target recognition, which can substantially speed up such neural-network algorithms.

Finally, Kulp[112] has studied the application of the backpropagation algorithm to the problem of automatic radar classification of surface ships. This work was performed on an SAIC Sigma-1 neurocomputer workstation that was obtained to facilitate insertion of neural-network technology into the applications domain. Figure 5 shows a display of the screen of the Sigma-1 after completion of the training phase. The network consisted of four layers. The first layer was the input data, which represented various bins of Fourier transformations of radar-profile data. The next two layers were the hidden units, and the last layer represented the output units that were compared to the desired classification outputs displayed in the target layer. The resultant error was backpropagated to adjust the weights. The unit values range from blue to red and represent the values of -5 to $+5$, respectively, that are input to the response function. Also plotted is the value of the error as a function of the iteration number.

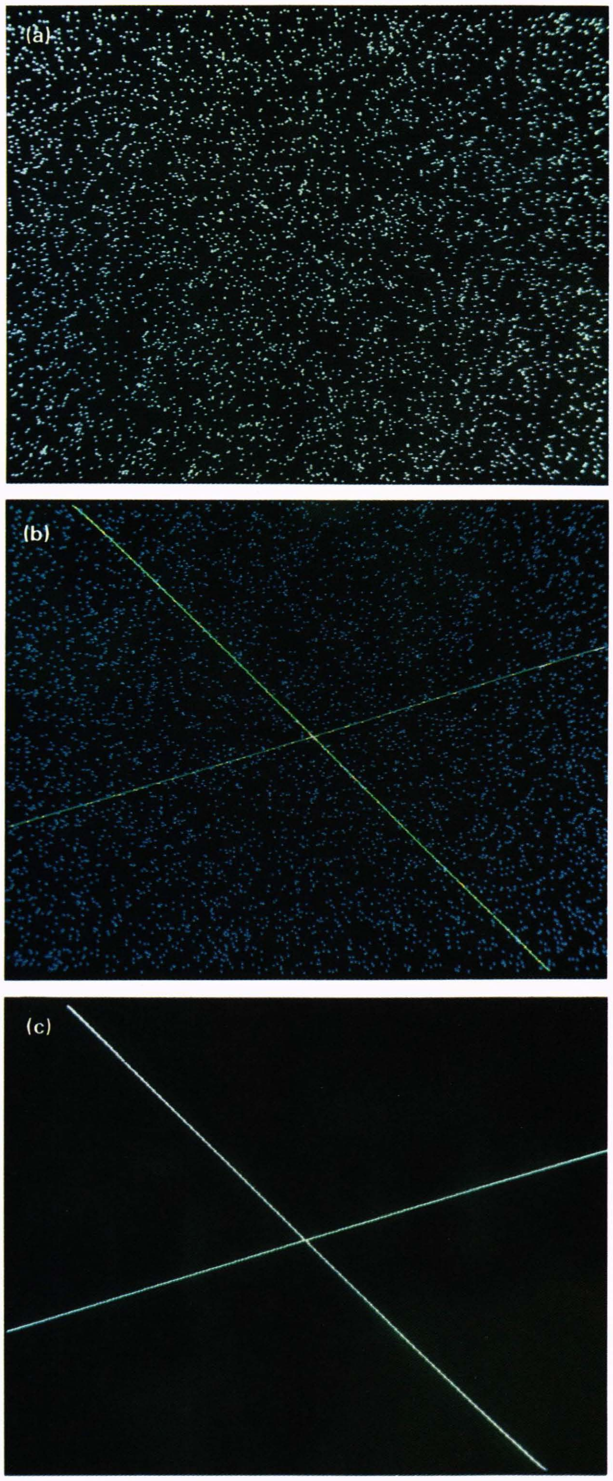
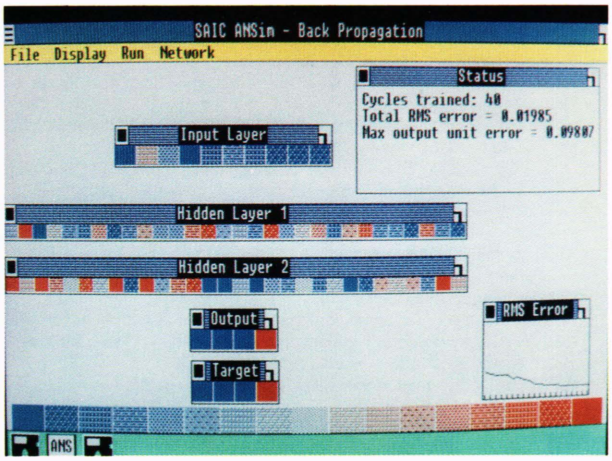


Figure 4—(a) Initial state of a two-track simulation with single-cell probabilities of detection and false alarm of 10 and 2%, respectively. (b) Two-track simulation using a modified Hopfield model after a few iterations and starting with Fig. 4a. (c) Final state of the two-track simulation using a modified Hopfield model starting with (a) and going through (b).[77]

THE FUTURE OF NEURAL-NETWORK TECHNOLOGY

Neural-network technology can resolve many problems that have resisted solution by traditional techniques.



```
Cycles trained: 40
Total RMS error = 0.01985
Max output unit error = 0.09807
```

Figure 5—Screen display of the SAIC Sigma-1 neurocomputer workstation for a radar target-recognition network developed using backpropagation.[112] (RMS error is the total root-mean-square error; ANSim is the software package; and ANS is an Artificial Neural System window.)

Such problems exist in the areas of signal processing, pattern recognition, control systems, and artificial intelligence. Neural-network technology also can produce a new kind of artificial intelligence that, instead of relying on the rules an expert might use to make decisions, learns from a series of examples. More intelligent knowledge bases can be built with neural-network technology. Through learning, neural networks could update the contents of a knowledge base and the heuristics used to process the knowledge. The full extent of the possibilities presented is still unknown.

Neural-network technology is not a replacement for existing computer technology. Investigators are currently working on understanding the potentials and limitations of neural-network technology; part of that effort includes learning how to combine both technologies into systems so as to make efficient use of their complementary abilities.

Finally, a wide variety of specialists have contributed to neural-network technology, including biologists, physicists, electrical engineers, computer scientists, and psychologists. Neural-network studies and applications are a multidisciplinary endeavor that will continue to benefit from the collaboration of such diverse specialists.

REFERENCES

[1]J. J. Hopfield and D. W. Tank, "Computing with Neural Circuits: A Model," *Science* **233**, 625–633 (1986).

[2]D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, Mass. (1986).

[3]R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Mag.*, pp. 4–22 (Apr 1987).

[4]G. E. Hinton, "Connectionist Learning Procedures," *Artif. Intell.*, in press (1988).

[5]D. J. Amit, "The Properties of Models of Simple Neural Networks," in *Heidelberg Colloquium on Glassy Dynamics,* J. L. van Hemmen and I. Morgenstern, eds., Springer-Verlag, New York, pp. 430–484 (1987).

[6]S. Grossberg, "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures," *Neural Networks* **1**, 16–61 (1988).

[7]J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Natl. Acad. Sci. USA* **79**, 2554–2558 (1982).

[8]J. J. Hopfield, D. I. Feinstein, and R. G. Palmer, "Unlearning Has a Stabilizing Effect in Collective Memories," *Nature* **304**, 158–159 (1983).

[9]J. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proc. Natl. Acad. Sci. USA* **81**, 3088–3092 (1984).

[10]J. J. Hopfield and D. W. Tank, " 'Neural' Computation of Decisions in Optimization Problems," *Biol. Cybern.* **52**, 141–152 (1985).

[11]D. W. Tank and J. J. Hopfield, "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," *IEEE Trans. Circuits Syst.* **CAS-33**, 533–541 (1986).

[12]J. J. Hopfield and D. W. Tank, "Collective Computation with Continuous Variables," in *Disordered Systems and Biological Organization,* E. Bienenstock et al., eds., Springer-Verlag, Berlin, pp. 155–170 (1986).

[13]D. W. Tank and J. J. Hopfield, "Neural Computation by Concentrating Information in Time," *Proc. Natl. Acad. Sci. USA* **84**, 1896–1900 (1987).

[14]D. W. Tank and J. J. Hopfield, "Concentrating Information in Time—Analog Neural Networks with Applications to Speech Recognition Problems," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. IV-455–IV-468 (1987).

[15]D. W. Tank and J. J. Hopfield, "Collective Computation in Neuronlike Circuits," *Sci. Am.* **257**, 104–114 (1987).

[16]D. Hebb, *The Organization of Behavior,* Wiley, New York (1949).

[17]D. J. Amit, H. Gutfreund, and H. Sompolinsky, "Spin-Glass Models of Neural Networks," *Phys. Rev. A* **32**, 1007–1018 (1985).

[18]D. J. Amit, H. Gutfreund, and H. Sompolinsky, "Statistical Mechanics of Neural Networks Near Saturation," *Ann. Phys.* **173**, 30–67 (1987).

[19]Y. S. Abu-Mostafa and J. St. Jacques, "Information Capacity of the Hopfield Model," *IEEE Trans. Inf. Theory* **IT-31**, 461–464, (1985).

[20]G. Weisbuch and D. D'Humieres, "Determining the Dynamic Landscape of Hopfield Networks," in *Disordered Systems and Biological Organization,* E. Bienenstock et al., eds., Springer-Verlag, Berlin, pp. 187–191 (1986).

[21]R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The Capacity of the Hopfield Associative Memory," *IEEE Trans. Inf. Theory* **IT-33**, 461–482 (1987).

[22]P. Baldi, "Symmetries and Learning in Neural Network Models," *Phys. Rev. Lett.* **59**, 1976–1978 (1987).

[23]K. F. Cheung, L. E. Atlas, and R. J. Marks, "Synchronous vs. Asynchronous Behavior of Hopfield's CAM Neural Net," *Appl. Opt.* **26**, 4808–4813 (1987).

[24]J. S. Denker, "Neural Network Models of Learning and Adaptation," *Physica* **22D**, 216–232 (1986).

[25]M. A. Cohen and S. Grossberg, "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks," *IEEE Trans. Syst. Man Cybern.* **SMC-13**, 815–826 (1983).

[26]T. Kohonen, *Self-Organization and Associative Memory,* Springer-Verlag, Berlin (1984).

[27]L. Personnaz, I. Guyon, and G. Dreyfus, "Neural Network Design for Efficient Information Retrieval," in *Disordered Systems and Biological Organization,* E. Bienenstock et al., eds., Springer-Verlag, Berlin, pp. 227–231 (1986).

[28]R. P. Lippmann, B. Gold, and M. L. Malpass, *A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification,* 769, MIT Lincoln Lab, Lexington, Mass. (1987).

[29]I. Kanter and H. Sompolinsky, "Associative Recall of Memory Without Errors," *Phys. Rev. A* **35,** 380–392 (1987).

[30]I. Guyon, L. Personnaz, P. Siarry, and G. Dreyfus, "Engineering Applications of Spin Glass Concepts," in *Heidelberg Colloquium on Glassy Dynamics,* J. L. van Hemmen and I. Morgenstern, eds., Springer-Verlag, New York, pp. 373–397 (1987).

[31]D. Horn and J. Weyers, "Hypercubic Structures in Orthogonal Hopfield Models," *Phys. Rev. A* **36,** 4968–4974 (1987).

[32]G. C. Fox and W. Furmanski, *Load Balancing by a Neural Network,* C^3P363, CALTECH, Pasadena, Calif. (1986).

[33]J. Barhen, N. Toomarian, and V. Protopopescu, "Optimization of the Computational Load of a Hypercube Supercomputer Onboard a Mobile Robot," *Appl. Opt.* **26,** 5007–5014 (1987).

[34]C. Koch, J. Marroquin, and A. Yuille, "Analog Neuronal Networks in Early Vision," *Proc. Natl. Acad. Sci. USA* **83,** 4263–4267 (1986).

[35]Y. C. Lee, G. Doolen, H. H. Chen, G. Z. Sun, T. Maxwell, H. Y. Lee, and C. L. Giles, "Machine Learning Using a Higher Order Correlation Network," *Physica* **22D,** 276–306 (1986).

[36]Ph. Lalanne, J. Taboury, and P. Chavel, "A Proposed Generalization of Hopfield's Algorithm," *Opt. Commun.* **63,** 21–25 (1987).

[37]E. Gardner, "Multiconnected Neural Network Models," *J. Phys. A* **20,** 3453–3464 (1987).

[38]J. D. Keeler, "Information Capacity of Outer-Product Neural Networks," *Phys. Lett. A* **124,** 53–58 (1987).

[39]C. L. Giles and T. Maxwell, "Learning, Invariance, and Generalization in High-Order Neural Networks," *Appl. Opt.* **26,** 4972–4978 (1987).

[40]D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Represen-

tations by Back-Propagating Errors," *Nature* **323**, 533-536 (1986).

[41]P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences,* thesis in applied mathematics, Harvard University, Cambridge, Mass. (1974).

[42]D. B. Parker, *Learning Logic,* Invention Report S81-64 File 1, Office of Technology Licensing, Stanford University, Stanford, Calif. (1982); also *Learning Logic,* TR-47, Center for Computational Research in Economics and Management Science, MIT (1985).

[43]Y. LeCun, "A Learning Procedure for an Asymmetric Threshold Network," *Proc. Cognitiva* **85**, 599-604 (1985).

[44]D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing, Explorations in the Microstructure of Cognition,* D. E. Rumelhart and J. L. McClelland, eds., MIT Press, Cambridge, Mass., pp. 318-362 (1986).

[45]F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms,* Spartan, Washington, D.C. (1961).

[46]M. Minsky and S. Papert, *Perceptrons,* MIT Press, Cambridge, Mass. (1969).

[47]S. E. Hampson and D. J. Volper, "Disjunctive Models of Boolean Category Learning," *Biol. Cybern.* **56**, 121-137 (1987).

[48]D. J. Burr, "A Neural Network Digit Recognizer," in *Proc. 1986 IEEE International Conf. on Systems, Man, and Cybernetics,* pp. 1621-1625 (1986).

[49]I. D. Longstaff and J. F. Cross, "A Pattern Recognition Approach to Understanding the Multi-Layer Perceptron," *Pattern Recognition Lett.* **5**, 315-319 (1987).

[50]B. Widrow, R. G. Winter, and R. A. Baxter, "Learning Phenomena in Layered Neural Networks," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-411-II-429 (1987).

[51]E. B. Baum, "On the Capabilities of Multilayer Perceptrons," CALTECH preprint, Pasadena, Calif. (1987).

[52]B. Widrow and M. E. Hoff, "Adaptive Switching Circuits," in *IRE Western Electronic Show and Convention,* Part 4, pp. 96-104 (1960).

[53]D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Sci.* **9**, 147-169 (1985); see also T. J. Sejnowski, P. K. Kienker, and G. E. Hinton, "Learning Symmetry Groups with Hidden Units: Beyond the Perceptron," *Physica* **22D**, 260-275 (1986).

[54]W. Y. Huang and R. P. Lippmann, "Comparisons Between Neural Net and Conventional Classifiers," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. IV-485-IV-492 (1987).

[55]G. E. Hinton, "Learning Distributed Representations of Concepts," in *8th Annual Conf. Cognitive Science Society*, Lawrence Erlbaum Associates, Hillsdale, N.J., pp. 1-12 (1986).

[56]E. Saund, "Abstraction and Representation of Continuous Variables in Connectionist Networks," in *Proc. AAAI-86 5th National Conf. on Artificial Intelligence,* Morgan Kaufmann, Los Altos, Calif., pp. 638-644 (1986).

[57]R. J. Williams, "Inverting a Connectionist Network Mapping by Back-Propagation of Error," in *8th Annual Conf. Cognitive Science Society,* Lawrence Erlbaum Associates, Hillsdale, N.J., pp. 859-865 (1986).

[58]E. B. Baum and F. Wilczek, "Supervised Learning of Probability Distributions by Neural Networks," CALTECH preprint, Pasadena, Calif. (1987).

[59]D. C. Plaut, S. J. Nowlan, and G. E. Hinton, *Experiments on Learning by Back Propagation,* CMU-CS-86-126, Carnegie-Mellon University, Computer Science Dept., Pittsburgh, Pa. (1986).

[60]C. R. Rosenberg and T. J. Sejnowski, "The Spacing Effect on NETtalk, a Massively-Parallel Network," in *Proc. 8th Annual Conf. Cognitive Science Society,* Lawrence Erlbaum Associates, Hillsdale, N.J., pp. 72-89 (1986).

[61]W. S. Stornetta and B. A. Huberman, "An Improved Three-Layer, Back Propagation Algorithm," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-637-II-643 (1987).

[62]E. D. Dahl, "Accelerated Learning Using the Generalized Delta Rule," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-523-II-530 (1987).

[63]J. P. Cater, "Successfully Using Peak Learning Rates of 10 (and Greater) in Back-Propagation Networks with the Heuristic Learning Algorithm,"in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-645-II-651 (1987).

[64]D. B. Parker, "Optimal Algorithms for Adaptive Networks – Second Order Back Propagation, Second Order Direct Propagation, and Second Order Hebbian Learning," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-593-II-600 (1987).

[65]R. L. Watrous, "Learning Algorithms for Connectionist Networks—Applied Gradient Methods of Nonlinear Optimization," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-619-II-627 (1987).

[66]A. Lapedes and R. Farber, "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modelling," preprint LA-UR-87-2662, Los Alamos National Laboratory, N.M. (1987).

[67]S. I. Gallant and D. Smith, "Random Cells—An Idea Whose Time Has Come and Gone ... and Come Again?", in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-671-II-678 (1987).

[68]R. S. Sutton, "Two Problems with Backpropagation and Other Steepest-Descent Learning Procedures for Networks," in *8th Annual Conf. Cognitive Science Society,* Lawrence Erlbaum Associates, Hillsdale, N.J., pp. 823-831 (1986).

[69]D. J. Volper and S. E. Hampson, "Connectionistic Models of Boolean Category Representation," *Biol. Cybern.* **54**, 393-406 (1986).

[70]G. E. Hinton, "Learning in Massively Parallel Nets," in *Proc. AAAI-86 5th National Conf. on Artificial Intelligence,* Morgan Kaufmann, Los Altos, Calif., p. 1149 (1986).

[71]S. E. Hampson and D. J. Volper, "Linear Function Neurons—Structure and Training," *Biol. Cybern.* **53**, 203-217 (1986).

[72]G. Tesauro and R. Janssens, *Scaling Relationships in Back-Propagation Learning: Dependence on Predicate Order,* CCSR-88-1, Center for Complex Systems Research, University of Illinois (1988).

[73]G. Tesauro, "Scaling Relationships in Back-Propagation Learning: Dependence on Training Set Size," *Complex Syst.* **1**, 367-372 (1987).

[74]S. Judd, "Learning in Networks Is Hard," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-685-II-692 (1987).

[75]A. Lapedes and R. Farber, "A Self-Optimizing, Nonsymmetrical Neural Net for Content Addressable Memory and Pattern Recognition," *Physica* **22D**, 247-259 (1986).

[76]F. J. Pineda, "Generalization of Backpropagation to Recurrent Neural Networks," *Phys. Rev. Lett.* **59**, 2229-2232 (1987); see also "Generalization of Backpropagation to Recurrent and High-Order Networks," in *Proc. IEEE Conf. on Neural Information Processing Systems—Natural and Synthetic* (1987).

[77]M. W. Roth, "Neural Networks for Extraction of Weak Targets from High Clutter Environments," JHU/APL F1A-1-87U-21 (1987).

[78]F. J. Pineda, "Dynamics and Architecture in Neural Computation," submitted to *J. Complexity* (1988).

[79]L. B. Almeida, "A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-609-II-618 (1987).

[80]R. Hecht-Nielsen, "Performance Limits of Optical, Electro-Optical, and Electronic Neurocomputers," in *Optical and Hybrid Computing* **634,** SPIE Conf. Proc., pp. 277-306 (1986).

[81]R. M. Kuczewski, M. H. Myers, and W. J. Crawford, "Neurocomputer Workstations and Processors—Approaches and Applications," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. III-487-III-500 (1987).

[82]R. Hecht-Nielsen, "Neurocomputing: Picking the Human Brain," *IEEE Spectrum* **25**(3), 36-41 (Mar 1988).

[83]C. R. Rosenberg and G. Blelloch, "An Implementation of Network Learning on the Connection Machine," Princeton University preprint (1987).

[84]A. P. Thakoor, A. Moopenn, J. Lambe, and S. K. Khanna, "Electronic Hardware Implementations of Neural Networks," *Appl. Opt.* **26**, 5085-5092 (1987).

[85]H. P. Graf, L. D. Jackel, R. E. Howard, B. Straughn, J. S. Denker, W. Hubbard, D. M. Tennant, and D. Schwartz, "VLSI Implementation of a Neural Network Memory with Several Hundreds of Neurons," in *Neural Networks for Computing, Snowbird, Ut. 1986, AIP Conf. Proc. 151,* American Institute of Physics, New York, pp. 182-187 (1986); see also L. D. Jackel, H. P. Graf, and R. E. Howard, "Electronic Neural Network Chips," *Appl. Opt.* **26**, 5077-5080 (1987).

[86]J. Alspector and R. B. Allen, "A Neuromorphic VLSI Learning System," in *Advanced Research in VLSI,* P. Losleben, ed., MIT Press, Cambridge, Mass., pp. 313-349 (1987).

[87]C. Mead, *Analog VLSI and Neural Systems,* Addison-Wesley, New York (1988).

[88]C. A. Mead and M. A. Mahowald, "A Silicon Model of Early Visual Processing," *Neural Networks* **1**, 91-97 (1988).

[89]Y. S. Abu-Mostafa and D. Psaltis, "Optical Neural Computers," *Sci. Am.* **256**(3), 88-95 (1987).

[90]T. Williams, "Optics and Neural Nets—Trying To Model the Human Brain," *Comput. Des.,* 47-62 (Mar 1987).

[91]D. Psaltis and N. Farhat, "Optical Information Processing Based on an Associative-Memory Model of Neural Nets with Thresholding and Feedback," *Opt. Lett.* **10**, 98-100 (1985).

[92]N. H. Farhat, D. Psaltis, A. Prata, and E. Paek, "Optical Implementation of the Hopfield Model," *Appl. Opt.* **24**, 1469-1475 (1985).

[93]G. D. Boyd, "Optically Excited Synapse for Neural Networks," *Appl. Opt.* **26**, 2712-2719 (1987).

[94]N. H. Farhat, "Optoelectronic Analogs of Self-Programming Neural Nets—Architecture and Methodologies for Implementing Fast Stochastic Learning by Simulated Annealing," *Appl. Opt.* **26**, 5093-5103 (1987).

[95]D. Psaltis, K. Wagner, and D. Brady, "Learning in Optical Neural Computers," in *IEEE 1st International Conf. on Neural Networks,* IEEE,

Piscataway, N.J., pp. III-549–III-555 (1987).

[96]D. Z. Anderson and D. M. Lininger, "Dynamic Optical Interconnects: Volume Holograms as Optical Two-Port Operators," *Appl. Opt.* **26**, 5031–5038 (1987).

[97]H. J. White, N. B. Aldridge, and I. Lindsay, "Digital and Analogue Holographic Associative Memories," *Opt. Eng.* **27**(1), 30–37 (1988).

[98]K. Wagner and D. Psaltis, "Multilayer Optical Learning Networks," *Appl. Opt.* **26**, 5061–5076 (1987).

[99]Y. Owechko, "Optoelectronic Resonator Neural Networks," *Appl. Opt.* **26**, 5104–5111 (1987).

[100]H. J. Caulfield, "Parallel N^4 Optical Interconnections," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. III-595–III-597 (1987).

[101]T. Higgens, "A Breakthrough for Optical Neural Nets," *Lasers and Optronics* **6**(11), 22–23 (1987).

[102]E. D. Dahl, "Neural Network Algorithm for an *NP*-complete Problem—Map and Graph Coloring," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. III-113–III-120 (1987).

[103]G. A. Tagliarini and E. W. Page, "Solving Constraint Satisfaction Problems with Neural Networks," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. III-741–III-747 (1987).

[104]J. D. Provence, "Neural Network Implementation for Maximum-Likelihood Sequence Estimation of Binary Signals in Gaussian Noise," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. III-703–III-714 (1987).

[105]N. H. Farhat, S. Miyahara, and K. S. Lee, "Optical Analog of Two-Dimensional Neural Networks and Their Application in Recognition of Radar Targets," in *Neural Networks for Computing, Snowbird, Ut. 1986, AIP Conf. Proc. 151,* American Institute of Physics, New York, pp. 146–152 (1986).

[106]R. M. Kuczewski, "Neural Network Approaches to Multi-target Tracking," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. IV-619–IV-633 (1987).

[107]L. D. Jackel, R. E. Howard, J. S. Denker, W. Hubbard, and S. A. Solla, "Building a Hierarchy with Neural Networks: An Example—Image Vector Quantization," *Appl. Opt.* **26**, 5081–5084 (1987).

[108]T. J. Sejnowski and C. R. Rosenberg, "NETtalk: A Parallel Network That Learns To Read Aloud," JHU/EECS-86/01 (1986); see also "Parallel Networks That Learn To Pronounce English Text," *Complex Syst.* **1**, 145–168 (1987).

[109]D. J. Burr, "Experiments with a Connectionist Text Reader," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. IV-717–IV-724 (1987).

[110]C. Priebe and D. Marchette, "An Application of Neural Networks to a Data Fusion Problem," in *Technical Proc. 1987 Tri-Service Data Fusion Symp.,* Naval Air Development Center, Warminster, Pa. (1987).

[111]R. P. Gorman and T. J. Sejnowski, "Analysis of Hidden Units in a Layered Network Trained To Classify Sonar Targets," *Neural Networks* **1**, 75–89 (1988).

[112]R. L. Kulp, "Initial Results and Observations from Investigation of Application of Neural Networks to a Ship Classification Problem," JHU/APL F1A-1-88U-004 (1988).

[113]G. W. Cottrell, P. Munro, and D. Zipser, *Image Compression by Back Propagation: An Example of Extensional Programming,* 8702, Institute for Cognitive Science, UCSD, La Jolla, Calif. (1987).

[114]R. M. Kuczewski, M. H. Myers, and W. J. Crawford, "Exploration of Backward Error Propagation as a Self-Organizational Structure," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-89–II-95 (1987).

[115]S. Shrier, R. L. Barron, and L. O. Gilstrap, "Polynomial and Neural Networks—Analogies and Engineering Applications," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-432–II-439 (1987).

[116]J. F. Shepanski and S. A. Macy, "Manual Training Techniques of Autonomous Systems Based on Artificial Neural Networks," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. IV-697–IV-704 (1987).

[117]A. Pellionisz and W. Graf, "Tensor Network Model of the 'Three-Neuron Vestibulo-Ocular Reflex-Arc' in Cat," *J. Theor. Neurobiol.* **5**, 127–151 (1987).

[118]D. Zipser and R. A. Andersen, "A Back Propagation Programmed Network That Simulates Response Properties of a Subset of Posterior Parietal Neurons," *Nature* **331**, 679–684 (1988).

[119]S. D. Post, "A Bipartite Connectionist Model to Represent N-ary Boolean and Linear Constraints," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. III-87–III-94 (1987).

[120]S. I. Gallant, "Connectionist Expert Systems," *Commun. ACM* **31**, 152–169 (1988).

[121]M. C. Mozer, "RAMBOT—A Connectionist Expert System That Learns by Example," in *IEEE 1st International Conf. on Neural Networks,* IEEE, Piscataway, N.J., pp. II-693–II-700 (1987).

[122]G. Tesauro and T. J. Sejnowski, "A Parallel Network That Learns to Play Backgammon," submitted to *Artif. Intell.* (1988).

[123]A. Bermar, "Firms Find More Uses for Neural Nets," *PC Week* **4**(6), C6 (1988).

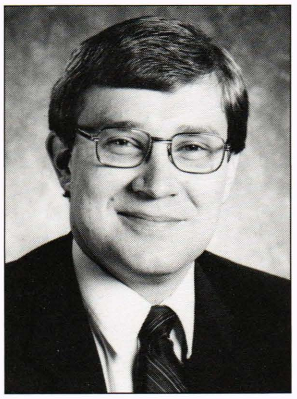
[124]T. J. Sejnowski, "Computational Models and the Development of Topographic Projections," *Trends Neurosci.* **10**, 304–305 (1987).

[125]J. M. Oyster, F. Vicuna, and W. Broadwell, "Associative Network Applications to Low-Level Machine Vision," *Appl. Opt.* **26**, 1919–1926 (1987).

[126]K. Strohbehn and A. G. Andreou, "A Neural Network Signal Detector Chip for the Search for Extraterrestrial Intelligence," JHU/APL preprint (1988).

ACKNOWLEDGMENTS—The author wishes to thank F. J. Pineda and R. L. Kulp for their comments and discussions on this paper.

THE AUTHOR



MICHAEL W. ROTH is a member of the Missile Systems Analysis Group at APL and has served as chief scientist, lead engineer, section supervisor, and project manager for numerous efforts at APL. He has also been a member of several internal and external committees such as the Advanced Technology Subcommittee of the APL Program Review Board. Dr. Roth was born in Davenport, Iowa, and received a B.A. degree in physics and mathematics from MacMurray College in 1971 and M.S. and Ph.D. degrees in physics from the University of Illinois in 1972 and 1975, respectively. He joined APL in 1977 after serving as a research associate at Fermilab. Dr. Roth has received numerous academic awards such as the Stuart S. Janney Fellowship, is a member of several professional societies such as the JHU chapter of Sigma Xi, has published papers on a variety of topics, has originated several inventions, and has been included in *American Men and Women in Science.*