Pattern Recognition: Neural Networks in Perspective

DeLiang Wang, Ohio State University

MOST NEURAL NETWORKS SHARE THE SAME STRENGTH AND WEAKNESS: GENERALIZATION OVER HAMMING DISTANCE. WHILE BREAKTHROUGHS IN PATTERN RECOGNITION CAN BE EXPECTED FROM NEURAL NET RESEARCH, THE KEY TO FUTURE SUCCESS WILL BE PROPER REPRESENTATIONS OF OBJECT STRUCTURE, NOT POWERFUL LEARNING ALGORITHMS. A NEW APPROACH, THE DYNAMIC LINK ARCHITECTURE, MAY HELP.

NEURAL NETWORK RESEARCH has grown explosively in the last decade, in areas ranging from the microscopic functioning of a single synapse to the macroscopic essence of consciousness, from solving the simple Exclusive Or mapping to intractable problems of weather forecasting. But across all problem domains, the greatest potential for neural net models is in pattern recognition, which causes little difficulty for people or even animals but is a staggering challenge to modern technology. Unlike a von Neumann machine, which sequentially executes a set of instructions, neural nets provide a new paradigm of computation based on networking many basic units and performing local computations in a massively parallel way, thus offering high hopes for pattern recognition.

The basic techniques of neural networks have been covered widely.[1-3] Instead of attempting a thorough review, this article focuses on four representative architectures with a unique thread: their ability to generalize. In my view, achieving proper generalization, or invariance, is the most challenging problem in pattern recognition, since mechanical template-matching techniques can easily produce stiff recognition.[4] Generalization over Hamming distance, a common approach, neglects the internal structures of objects to be recognized, thus severely limiting applications of most neural net models to invariant pattern recognition. The dynamic link architecture is a new approach, promising to overcome some of the difficulties facing classical neural net models. Representation and invariance are two key issues for all such models doing pattern recognition.

Associative memories

Associative (or content-addressable) memories have been a key area of neural net research.[5-7] The correspondence between associative memory and pattern recognition is evident. Associative memory models can be classified into autoassociative and heteroassociative. The major feature of autoassociative models is pattern completion, that is, the ability to recall a stored pattern (template) when a similar pattern is presented. Heteroassociative models build associations between pairs of patterns. This article focuses on one type of autoassociate model, the Hopfield model, which is widely regarded as a major impetus to the current resurgence of interest in neural networks. In terms of pattern recognition, heteroassociative models resemble the mapping network reviewed later.

The basic idea. Hopfield nets use an intuitive idea: They store each pattern (template) as connections between components. When a new input pattern is presented, the associative memory activates (recalls) the

0885/9000/93/0800-0052 $3.00 © 1993 IEEE

stored pattern that most closely resembles the input. Due to the positive and negative connections, the network can recall an entire pattern on the basis of a sufficient subpart.

System architecture. The Hopfield architecture is simple: a single-layer network with recurrent connections (see Figure 1).[7] A network has N units of V_i ($i = 0, ..., N-1$). V_i takes a binary value (Hopfield specified $-1/1$ values, but it is easy to convert a 0/1 binary-valued x into $-1/1$ value by $2x-1$). As an important constraint, connection weights must be symmetrical; that is, the same weight is assigned to both directions between two units. Each unit connects to all other units in the network. Each stored pattern $\mathbf{P}^r$ ($r = 1, ..., M$) relates to a vector of N binary ($-1/1$) components, corresponding to a state of the network.

Algorithm. The algorithm for Hopfield nets is as follows:

(1) Assign connection weights:

$$w_{ij} = \begin{cases} \sum_{r=1}^{M} P_i^r P_j^r, i \neq j & 0 \leq i, j \leq N-1 \\ 0, \quad i = j \end{cases} \quad (1)$$

where w_{ij} is the connection from unit j to unit i. The weight assignment is done only once as the patterns are stored, and no synaptic modification occurs as the network computes.

(2) Present an input pattern $\mathbf{I}$ by assigning the initial state of the network as the input pattern: $V_i(0) = I_i$.

(3) To compute the network, update the states of the network units asynchronously:

$$V_i(t+1) = sgn\left[\sum_{j=1}^{N-1} w_{ij} V_j(t)\right] \quad (2)$$

where the function $sgn(x) = 1$ if $x \geq 0$, or -1 if $x < 0$.

(4) Repeat updating by going to step 3 until equilibrium is reached, that is, until no V_i changes between iterations. At equilibrium, the vector of V_i's is the pattern recalled by input $\mathbf{I}$.

(5) To continue with the next input pattern, go to step 2.

Related results. In the same paper, Hopfield introduced the idea of an energy function (generally known as the Lyapunov function in dynamical systems theory), defined as

$$H = -\frac{1}{2}\sum_{ij} w_{ij} V_i V_j \quad (3)$$

and emphasized the idea of viewing stored patterns as dynamical attractors (minima of the energy function).[7] On the basis of the energy function, he showed that the network converges when the units are updated asynchronously using Equation 2, and also converges with analog unit values instead of binary values.[8] Later, Hopfield and Tank used the energy function to solve small versions of the NP-complete Traveling Salesman Problem,[9] triggering a great deal of optimization research.

The Hopfield model has two major limitations. First, its capacity (the number of patterns that can be stored and accurately recalled) is severely limited. An optimistic estimate of the maximum capacity is only about $0.138N$ for random patterns, far less than the number of possible states 2^N. Second, overlapping between stored patterns can cause the network to reach an equilibrium not corresponding to any stored pattern. This phenomenon is called the spurious local minimum problem. Boltzmann Machine networks can overcome this problem using a simulated annealing procedure governed by Boltzmann statistics.[10] Imposing constraints on the stored patterns, such as orthogonalization, might also alleviate the problems of capacity and spurious local minima.

Relation to pattern recognition. When the Hopfield model is used as a recognizer, the network activity after convergence is compared with stored templates, and the template that best matches the output is selected. This template is the result of recognition if it sufficiently matches the network output; otherwise, a "no match" result occurs. Because associative memory models can be used for pattern completion, the Hopfield net can generalize over Hamming distance (the number of different bits or pixels between two binary patterns) when used as a pattern recognizer. That is, in the ideal situation, the model recalls the stored template that has the smallest Hamming distance with a given input pattern.

Backpropagation

The backpropagation algorithm had a colorful beginning,[11] and its development

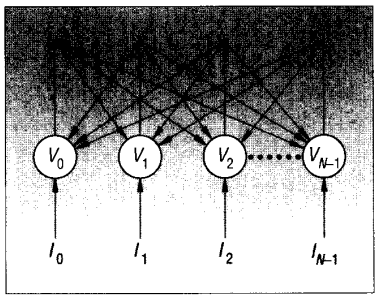

Figure 1. Hopfield net for associative memory. Each unit connects to all other units in the network.

is one of the most important advances in the past decade of neural computation. The description here follows the widely known version by Rumelhart, Hinton, and Williams,[12] although it is not considered the first version. The backpropagation network is a multilayer feedforward network that uses the backpropagation training algorithm. It is often called a multilayer perceptron because it represents a significant extension to the classic perceptron model by Rosenblatt,[13] whose learning rule applies only to a single-layer network, and whose samples to be classified must be linearly separable. For example, the simple perceptron model cannot compute the Exclusive Or of two binary inputs. Although people realized that adding more layers could enhance capacity, appropriate learning rules were absent for multilayer perceptrons until backpropagation was invented. By introducing a hidden layer between the input and output layers (see Figure 2), the backpropagation network provides a powerful learning algorithm that has been applied to a wide variety of applications.

The basic idea. Backpropagation typifies supervised learning, where the task is to learn to map input vectors to desired output vectors. The backpropagation learning algorithm modifies feedforward connections between the input and the hidden units, and the hidden and output units, so that when an input vector is presented to the input layer, the output layer's response is the desired output vector. During training, the error caused by the difference between the desired output vector and the output layer's response to an input vector propagates back through connections between layers and adjusts appropriate connection weights (credit assignments) so as to minimize the error.

System architecture. The three-layer

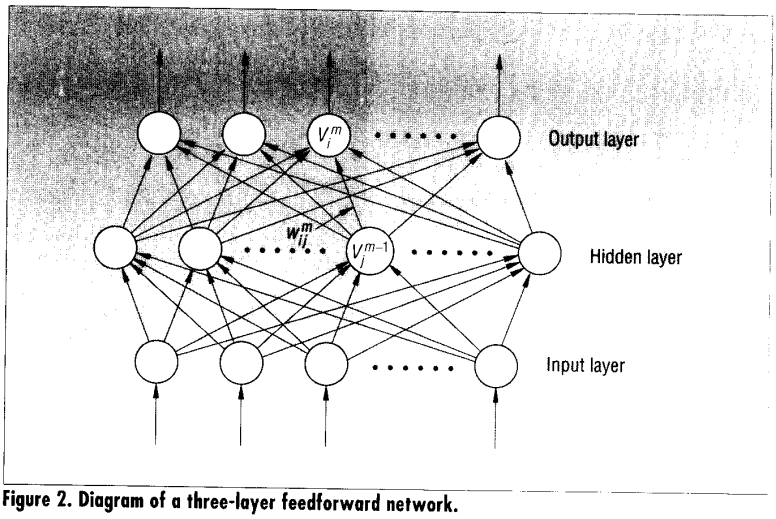

Figure 2. Diagram of a three-layer feedforward network.

network shown in Figure 2 is typical. The learning algorithm works for any M+1-layer network, thanks to the chain rule of differentiation. V_i^m represents the output of the ith unit of layer m, and takes analog values usually generated by a sigmoid function of the form

$$f(x) = \frac{1}{1+e^{-(x-\theta)}} \quad (4)$$

with θ as the bias or threshold for the unit.

Algorithm. The backpropagation algorithm is as follows:

(1) Set all weights to small random values.
(2) Present an input vector **I** and a desired output vector **O**. Apply **I** to the input layer ($m = 0$) so that $\mathbf{V}^0 = \mathbf{I}$.
(3) For other layers, namely $m = 1, \ldots, M$, perform forward computation:

$$V_i^m = f\left(\sum_j w_{ij}^m V_j^{m-1}\right) \quad (5)$$

where w_{ij}^m represents the connection weight from V_j^{m-1} to V_i^m.
(4) Compute the errors for the output layer:

$$\delta_i^M = V_i^M\left(1-V_i^M\right)\left(O_i - V_i^M\right) \quad (6)$$

(5) Compute the backpropagated errors for preceding layers $M-1, \ldots, 1$:

$$\delta_i^{m-1} = V_i^{m-1}\left(1-V_i^{m-1}\right)\sum_j w_{ji}^m \delta_j^m \quad (7)$$

(6) Adjust all weights:

$$w_{ij}^m(t+1) = w_{ij}^m(t) + \eta\delta_i^m V_j^{m-1} \quad (8)$$

where η is a gain parameter. Thresholds are adjusted in a way similar to weights.
(7) Repeat by going to step 2.

The algorithm continues until the overall error, which is the mean square difference between the desired and the actual outputs for all training patterns, is reduced to an acceptable level.

Related work. The learning rule in this algorithm reduces the overall error function E by gradient descent. (This method reduces E by modifying a weight along the reverse direction of the slope of E with respect to the weight; namely, ΔW_{ij} is proportional to $-\partial E/\partial W_{ij}$, which is the sum of the squared distance between the actual output response to an input and the desired output. See Equation 18, given later, for an example.) Gradient-descent methods, such as backpropagation, are plagued by the local-minima problem. However, for reasons that are not really understood, the scheme has not suffered much from local minima, as shown in many simulations. It has also been proven that one hidden layer (Figure 2) is sufficient to approximate any mapping function,[14] although with an excruciatingly large number of hidden units. Other researchers have extended backpropagation to train recurrent networks in continuous time.[15-16]

Besides the practical concern that training is generally slow (more on this shortly), one of the algorithm's major theoretical obstacles is that formal analysis has been difficult, if not impossible. For example, why does the hidden layer develop "sensible" internal representations, crucial to the success of training? Nonetheless, this network model has enjoyed great success in solving various problems.

Relation to pattern recognition. When the backpropagation network is used for pattern recognition, nodes in the output layer usually correspond to stored templates. Different from the associative memory models where patterns are directly assigned to a network, the backpropagation architecture needs an extensive training session to establish stored templates. Afterwards, the network can be tested on the training as well as on new examples. For instance, the algorithm has been applied to recognizing handwritten zip code digits: 7,291 examples were used as the training set, and another 2,007 were used for testing generalizations.[17] The correct percentage on the test set was 95 percent, an impressive result.

During training, the high-dimensional input space is divided into category regions, each of which forms the same classification decision. The significant contribution of the backpropagation algorithm is that it can form arbitrarily complex decision regions, which is not a property of the simple perceptron. A category region is formed based on extrapolation of its training examples, and it by itself need not be a connected region. Thus, if training is successful, the category region will include all examples belonging to the category but no example belonging to any other category. New patterns are classified based on which category region they fall in. Since the formation of a decision region is based on extrapolation in the input space, where the distance between two points can be measured as the Hamming distance, the generalization ability of backpropagation is to a certain extent also based on the Hamming distance between the input pattern to be classified and the training examples. Roughly speaking, the input pattern is classified into the same category as its neighbors in the example space, if they are sufficiently close.

The ART architecture

Carpenter and Grossberg developed a neural architecture for pattern recognition, called the adaptive resonance theory. ART

has gone through three stages: ART 1,[18] ART 2,[19] and ART 3.[20] Although each later version enhances ART capabilities in certain ways, this review will only consider ART 1, since it possesses the architecture's basic design philosophy.

The basic idea. ART is a type of competitive learning network, and suitable for both category (pattern) formation and recall (recognition). When an input pattern is sufficiently similar to ("resonates" with) one of the stored categories (represented by single units), ART recognizes the pattern as belonging to the category, and modifies the category itself to accommodate new features of the current input pattern. When an input pattern is not sufficiently similar to any stored category, pattern formation occurs: ART selects an uncommitted unit to code the current input. If no more uncommitted nodes are left, the current input results in no response. Thus, stored categories remain stable to irrelevant input events, and yet are sensitive to novel features of inputs. That is, ART makes a satisfactory trade-off between stability and plasticity.

System architecture. Figure 3 shows a simplified version of ART 1. It is a network of two layers, F_1 and F_2, which are fully connected in both directions. The forward and backward connections between F_1 and F_2 contain memory traces. F_2 also contains category units that connect with each other, implementing the winner-take-all function (maximum selector). Each category unit can be enabled or disabled. The input component I_i, the backward weight z_{ij}, as well as units U_i, V_j, A, and R all take binary 0/1 values. Unit A enables F_1 to distinguish between a top-down recalled template and the bottom-up input. Unit R resets the active F_2 unit if mismatch occurs between the stored pattern and the input.

Algorithm. The following algorithm implements the basic idea of ART:

(1) At time 0, set $z_{ij}(0) = 1$ and $w_{ji}(0) = 1/(\varepsilon+N)$, where $0 \le i \le N-1$ and $0 \le j \le M-1$. The small ε breaks ties. Enable all units in F_2, and set the vigilance threshold ρ, which determines how close an input pattern must be matched to a stored template.

(2) Apply a new input vector $\mathbf{I}$.

(3) Find the unit that is best activated by the current input. This is done by comparing the inner product $\mathbf{I}\cdot\mathbf{w}_j$ ($0 \le j \le M-1$) of all the enabled units of F_2. The winner V_J has the greatest value $\mathbf{I}\cdot\mathbf{w}_J$. This is implemented by the winner-take-all network built in F_2. The overall outcome is $V_J = 1$, $V_j = 0$ for $j \ne J$.

(4) To match $\mathbf{I}$ and the recalled template $\mathbf{z}_{\cdot J}$ (the vigilance test), compute the ratio

$$r = \frac{\sum_i I_i z_{iJ}}{\sum_i I_i} \qquad (9)$$

If $r \ge \rho$, there is resonance; go to step 5. Otherwise, disable V_J and repeat the process of finding an appropriate category by going to step 3.

(5) Adjust the winning vector $\mathbf{z}_{\cdot J}$ by performing a logical And operation. This deletes any bits in it that are not also in $\mathbf{I}$ (weight $\mathbf{w}_j$ is a normalized version of $\mathbf{z}_{\cdot j}$, and thus is adjusted too):

$$z_{iJ}(t+1) = z_{iJ}(t) \ll I_i \qquad (10)$$

$$w_{Ji}(t+1) = \frac{z_{iJ}(t+1)}{\varepsilon + \sum_i z_{iJ}(t+1)} \qquad (11)$$

(6) Repeat by going to step 1.

Steps 3 and 4 of this algorithm form the search process. In particular, let's look at how auxiliary units A and R achieve their functions in a desirable way. The units U_i are designed so that

$$U_i = \begin{cases} I_i & \text{if no } V_i \text{ is on} \\ I_j \cap \sum_j z_{ij} V_j & \text{otherwise} \end{cases} \qquad (12)$$

Since only one unit can be activated at a time in F_2, the second part of Equation 12, which is used for the vigilance test, takes only binary values. The choice here can be achieved with unit A, which is triggered when there is an input but no unit in F_2 is activated. Thus, A is equal to $[\sum_i I_i - N \sum_j V_j]^+$, where the step function $[x]^+ = 1$ if $x > 0$, or 0 otherwise. Let U_i receive the input

$$h_i = I_i + \sum_j z_{ij} V_j + A \qquad (13)$$

and $U_i = [h_i - 1.5]^+$. This formation of U_i is equivalent to Equation 12, and is referred to as the 2/3 rule, since two out of the three inputs I_i, $\sum_j z_{ij} V_j$, and A must be on to activate U_i.

The disabling signal (Figure 3) is generated when r in Equation 9 is less than ρ. This is equivalent to letting $R = [\rho \sum_i I_i - \sum_i U_i]^+$.

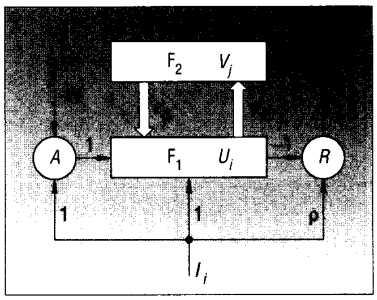

Figure 3. Diagram of the ART 1 architecture. The thick arrows represent connections between all units U_i in F_1 and all units V_j in F_2.

Related work. While ART 1 is designed for binary inputs, ART 2 can take either analog (continuous-valued) or binary inputs.[19] Also, in ART 1, a stored template will gradually diminish by the And operation of Equation 10 if elements of input patterns are missing due to noise. ART 2 has overcome this gradual diminishing problem by replacing the And operation with a gradual tuning that adjusts the stored template to the current input pattern. By introducing plausible neurotransmitter modulation, ART 3 justifies the disabling process in ART 1 and 2 that is due to a mismatch (see Figure 3). Also, previous asymmetry between layers F_1 and F_2 is no longer necessary in ART 3,[20] thus allowing a hierarchy of layers. One potential problem with ART networks is the serial search time, which in the worst case is proportional to the entire capacity of long-term memory. When storing a large number of items, this problem is a serious concern.

Relation to pattern recognition. Primarily designed for pattern recognition, ART has been applied to such problems as visual pattern recognition, speech perception, and radar classification.[21] For example, when ART 2 was used to classify analog curves, the network classified 50 patterns into 34 categories,[19] and results appeared very reasonable. As clearly shown in Equation 9, ART classifies patterns on the basis of Hamming distance, and thus bases generalization on the same measurement used by the other models already discussed.

Dynamic link architecture

The idea of dynamic links was first proposed by von der Malsburg in 1981 as a theory of brain function,[22] which he later

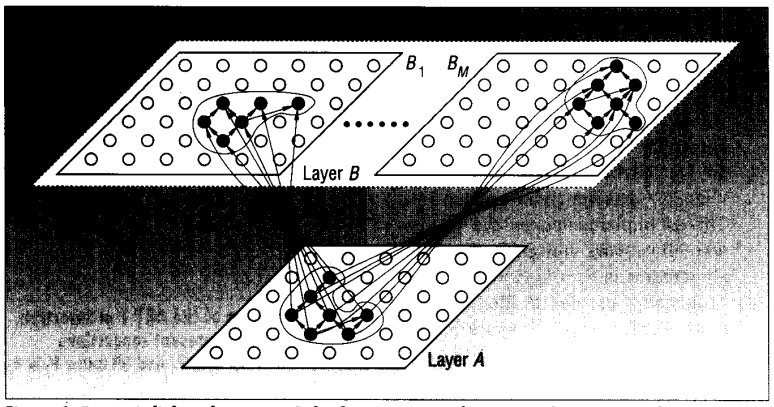

Figure 4. Dynamic link architecture. Only the connections between relevant units that are occupied by some pattern are shown (in black).

condensed into a paper.[23] He challenged a fundamental doctrine of neurobiology and neural network research: that neurons (units) code features, while connections between neurons code memory. Neuron activities are updated rapidly, corresponding to feature onset and offset, but connection weights are modified slowly, corresponding to long-term memory. All three neural net frameworks reviewed earlier adopt this view.

According to the doctrine, the presentation of a green A activates two units corresponding to features "green" and "A." Thus, if a green A and a red B are presented at the same time, four units corresponding to "green," "red," "A," and "B" are all activated. Now the question is, how can the brain distinguish "green A" and "red B" as actually presented versus "green B" and "red A"? There is no way to avoid this confusion by looking at the activities of the four units. To get out of this difficulty, many models introduce high-level units that code combinations of features, leading to the notion of *grandmother cells*. Besides other problems such as reliability, this scheme would result in a formidable unit population, since two units are required to code both the smiling grandmother and the crying grandmother.

To solve this problem, von der Malsburg proposed dynamic links, in which, in addition to unit activity, connections between units are also rapidly modified, corresponding to the onset and offset of patterns. Thus, there are two connection weights between any two units for coding memories: the *temporary link* for fast changes, and the *permanent link* for slow changes. Back to the previous example, when the green A and the red B are presented at the same time, the link between features "green" and "A" and the link between "red" and "B" are also activated. This regulates the activity of the four feature units so that there is correlation of firing between units "green" and "A," and between "red" and "B." (One way to implement this is to use phase locking between firing impulses, as shown in Figure 5 later.)

The basic idea. I will skip the theory's biological implications and concentrate on the dynamic link architecture for pattern recognition. Pattern recognition must be invariant under various transformations; for visual pattern recognition, this includes translation, dilation, rotation, and distortion (possibly due to changes in viewing perspectives). To achieve invariant pattern recognition, a network must explicitly encode neighborhood or topological relations between a pattern's features. Dynamic links do just that: they are activated, like neurons, in response to the occurrence of specific topological relations. Therefore, instead of having neuronal dynamics, we now have connection dynamics. Recognition corresponds to a graph-matching process: Input patterns and stored patterns are viewed as graphs, rather than sets of unit onsets.

System architecture. Different versions of the dynamic link architecture for invariant pattern recognition have been proposed in the last few years. I will describe a simplified version of earlier work:[24-25] The architecture consists of two layers, an input layer A and a memory layer B (see Figure 4). The memory layer consists of identical M subnets $B_1,...,B_M$, each of which holds one stored pattern. Layers A and B_r ($r = 1, ...,M$) are 2D grids of units, and A fully connects to each B_r. Within all grids, each unit V_{ij} connects only to its three neighbors $V_{i+1,j-1}$, $V_{i+1,j}$, and $V_{i+1,j+1}$, and the subnets in B are not connected to each other. The connection from node V_{kl} to node V_{ij} is described by two types of weights: a permanent binary 1/0 link $w_{ij,kl}$, and a temporary link $z_{ij,kl}$ that satisfies

$$0 \le z_{ij,kl} \le w_{ij,kl} \tag{14}$$

Assume that each stored pattern $\mathbf{P}^r$ ($r = 1,...,M$) is a matrix of binary components.

Algorithm. The algorithm for the dynamic link architecture is as follows:

(1) Assign permanent weights. Each pattern $\mathbf{P}^r$ to be stored is assigned to subnet B^r such that

$$w_{ij,kl} = \begin{cases} P^r_{ij} P^r_{kl} & \text{if } i = k+1,\ j = l-1, l, \text{or } l+1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

The long-term weight assignment in B is done only once as the patterns are stored. In addition, $w_{ij,kl} = 1$ if $V_{ij} \in B$ and $V_{kl} \in A$.

(2) Present an input pattern $\mathbf{I}$ to A by assigning temporary links within A:

$$z_{ij,kl} = \begin{cases} I_{ij} I_{kl} & \text{if } i = k+1,\ j = l-1,\ l, \text{or } l+1 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

After this is done, the temporary links in A are fixed (clamped) until the recognition process is finished. Also, initialize $z_{ij,kl} = \varepsilon\, w_{ij,kl}$ for $V_{ij} \in B$ and $V_{kl} \in A$ or B, where ε is a random value between 0 and 1.

(3) Match an input graph with a stored one by minimizing this energy function:

$$\begin{aligned} H(z) = &- \sum_{i,j \in B; k,l \in A} z_{ij} z_{jl} z_{ik} z_{kl} + \\ &\gamma \sum_{i \in B} \left(\sum_{k \in A} z_{ik} - 1 \right)^2 + \\ &\gamma \sum_{k \in A} \left(\sum_{i \in B} z_{ik} - 1 \right)^2 \end{aligned} \tag{17}$$

Then minimize $H(z)$ using gradient descent:

$$z_{ij}(t+1) = \left[z_{ij}(t) - \eta \frac{\partial H(z(t))}{\partial z_{ij}(t)} \right]^w \tag{18}$$

where $[\cdot]^w$ means that $z_{ij}(t+1)$ is confined to the interval $[0, w_{ij}]$. At equilibrium, $H(z)$ is minimized, and the connection pattern on layer B represents the pattern recalled by input $\mathbf{I}$.

(4) Repeat by going to step 2.

By minimizing the energy function in Equation 17, the algorithm retrieves a specific stored pattern that is similar in structure to the current input pattern. The first term in Equation 17 plays the role of mapping the edges of the graph corresponding to the input pattern on layer A onto graphs stored in the subnets of layer B. Since the subnets of layer B are not linked, this term favors neighborhood correlation between the input pattern and each stored pattern on a subnet. The second and third terms provide constraints on the number of links for the mapping from A to B: Each node in A should map to one and only one node in B. Taken together, in low-energy states, 1-to-1 mapping from A to B will be established.

Related work. Pattern recognition by graph matching is essentially equivalent to the problem of finding graph isomorphism, in general an NP-complete problem.[26] However, von der Malsburg argued that problems arising from planar graphs that are used mostly for practical 2D pattern recognition are computationally tractable.[25] As shown elsewhere,[24] pattern recognition in this scheme achieves translation invariance and moderate distortion invariance. A certain amount of overlapping in layer B is also tolerable, such that the number of patterns stored in B can be smaller than the number of subnets. Gradient descent as used in Equation 18 is a heuristic method, and can be replaced by other techniques such as simulated annealing.[27]

The major restriction of this architecture is that rigorous formal analysis is not available. For example, no proof is given whether the network system always converges, and if so, whether the energy minimum is a global one. Many other practical questions, such as the speed of convergence and storage efficiency, remain unanswered. The lack of formal treatment clouds the assessment of the dynamic link architecture's usefulness.

Relation to pattern recognition. Like the Hopfield model, when the dynamic link architecture is used as a recognizer, the $\{z_{ij}\}_{i,j\in B}$ after convergence is compared with templates stored in B (that is, $\{w_{ij}\}_{i,j\in B}$), and the best-matched template is selected. If this template is sufficiently close to the current connection pattern on B, the template is the output of the recognition process; otherwise, "no match" is the output. Researchers are applying this approach to human-face recognition.[28-29] A portrait gallery of 87 face images was stored, and patterns resulting from different facial expressions, viewing perspectives, and sizes of stored templates were used as the test set. The overall error was less than 15 percent.[29]

Invariant pattern recognition is the motivation behind the dynamic link architecture. The algorithm provides translation invariance, since the connections from A to B are not specific to locations of layer A. Also, pattern recognition is to a certain degree invariant to systematic (as opposed to random) distortions such as changes in perspective, size, and so on. Distortion can change a pattern drastically in terms of Hamming distance, but the change appears much less drastic if viewed from the perspective of connection patterns; neighboring nodes in a graph still tend to be neighbors after systematic distortions. In other words, the algorithm demonstrates certain invariance to "rubber-sheet" transformations. This is because pattern recognition is based on image structures (graph isomorphism) rather than eidetic images (Hamming distance).

Other architectures

As stated in the beginning, I have not tried to give an extensive review, thus necessarily overlooking many other neural net architectures. For instance, in Kohonen's self-organizing map,[30] trained units spatially near each other in an output layer recognize similar input patterns. This unique feature enables the network to perform well in noisy data. But again, this approach, like ART, measures similarities between patterns by Hamming distance.

Representation and invariance

The neural net approach provides a nontraditional way of formulating and solving problems, and most popular neural networks perform as well as traditional pattern recognition approaches.[30] However, it is too early to say that current techniques provided by neural networks have outperformed traditional algorithms. As mentioned earlier, pattern recognition in most neural networks generalize over Hamming distance, a property achieved long ago.[4] Although networks possess a capacity for learning, if learning cannot augment performance, its value inevitably will be depreciated. We could argue that neural algorithms are inherently parallel; however, most traditional pattern recognition algorithms are also parallel, so both will exhibit increased performance on parallel machines. To strengthen its position in the toolbox of pattern recognition designers, the field of neural networks will eventually need to demonstrate better performances than traditional approaches.

These comments are not intended to downplay other important contributions of neural networks. For example, neural computation has an important characteristic: its uniform structure is composed of only units and connections. With the backing of massively parallel VLSI technology, this uniform structure might allow neural networks to become a new-generation non-von Neumann computing architecture.[32] Another important feature of the neural net approach is its robustness to noise. This is due partly to distributed representation and ample interconnections built naturally into the network. Also, the methodology of neural computation appeals to neuroscientists. In this respect, biological plausibility becomes the top assessment criterion of a model. To enhance understanding of brain functions, neural models must explain observed data and predict new phenomena subject to experimental testing (an example study can be found elsewhere[33]).

Invariance and Hamming distance. Invariance is a fundamental property of pattern recognition. However, basic types of invariance such as translation, rotation, scale, and various distortions are beyond generalization over Hamming distance. Thus, neural networks that use this measure to generalize can deal only with very limited invariance. We might argue that a backpropagation network can be trained to discriminate a triangle and a circle without reference to their positions, given sufficient training examples. The nature of supervised learning is example-based, and the failure of performance can always be

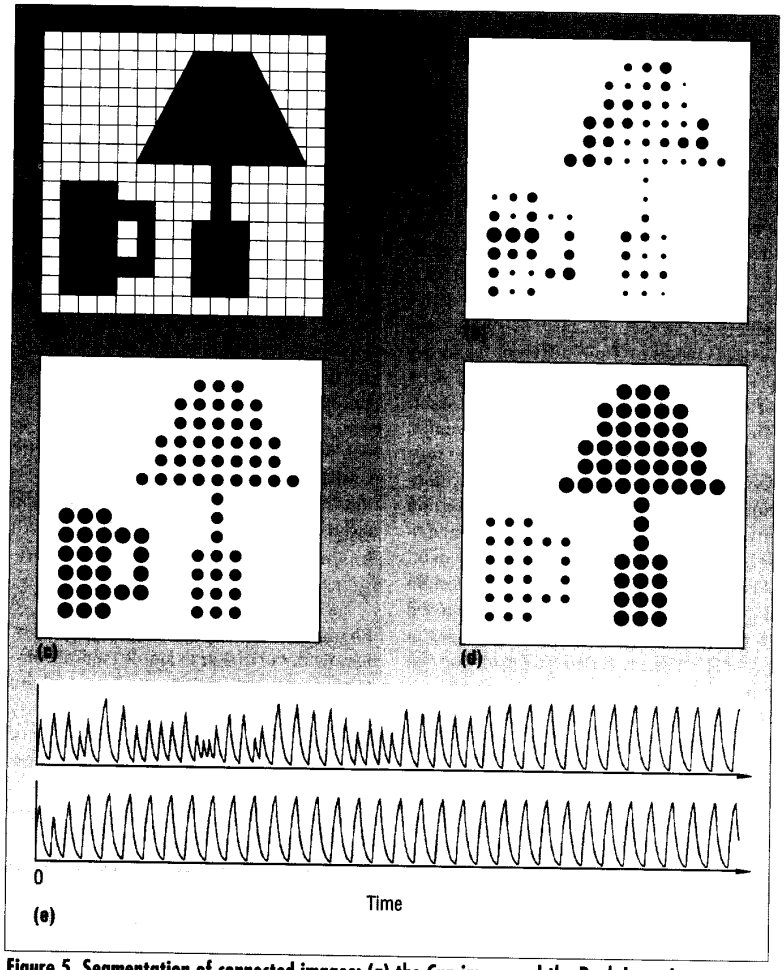

Figure 5. Segmentation of connected images: (a) the Cup image and the Desk Lamp image are presented to a 15x15 grid of oscillators; (b) the instant activity of the oscillator grid at the beginning of dynamic evolution (the amplitude of each oscillator's activity is indicated by the circle's diameter; only the oscillators whose activities are nonzero are shown); (c) and (d) the network activity at two time instants after the system has evolved for a short time; (e) the temporal activities of two typical oscillators representing the two images.

attributed to inadequate examples. But if all possible examples are given in training, there is no generalization at all. In addition, the number of examples needed for learning a nontrivially few concepts would be prohibitively large, and this is becoming an increasing concern.[34]

Many models, such as ART, treat invariance as part of preprocessing and resort to techniques developed in computer vision.[21] Invariance over linear transformations (translation, rotation, scale) might be achievable via a Fourier transform or similar transforms, which can be implemented by proper neural network architectures.[35] But the invariance of distortion, for instance, a flag swinging in the wind, can in no way be achieved by such transforms.

Invariance, in my view, will eventually be captured in neural networks by proper representations, not by powerful learning algorithms. Various types of invariance reflect the constructions of the world and the way we interact with it. For example, the same object in the real world can be perceived from different angles (translation), distances (scale), and perspectives (rotation); the same object might be deformed (distortion) by nature, such as growth or wind. Moreover, the object might be situated in a complex background with many other objects, and it must be segregated from the background before it can be perceived properly. The brain must recognize these transformations to survive (to catch prey or avoid predators, for example), and this general-purpose ability should be inherited through proper representations. Thus, a major challenge is how to encode correct constraints into neural networks before they are subject to learning. The following case study illustrates this point.

Segmentation of connected images. Segmentation is an important stage of perception and a critical preprocessing step for recognizing specific patterns. It is the inverse of perceptual grouping: forming a single segment. According to Gestalt psychology, one of the most important principles of grouping is connectedness, the strong tendency of the visual system to group a connected region as one segment.[36] Perceptual grouping is largely innate. The following network for segmenting connected images is based on my recent finding that the dynamics of a network of locally coupled neural oscillators can drive the network to reach phase synchrony.[37] Segmentation is based on the idea that a single segment is expressed by phase synchrony within its constituent components (pixels), and different objects are expressed by different phases of oscillations.

A 2D network is constructed for this task, with each oscillator connecting locally to its four neighbors, thus forming a grid. Suppose we want to segment the image of a cup from that of a desk lamp. Figure 5a shows a grid of 15×15 units and the images. Apparently, Cup and Desk Lamp are both connected regions, but they are not connected to each other. Each oscillator i is defined as a feedback loop between an excitatory unit x_i and an inhibitory unit y_i:

$$dx_i/dt = -x_i + f_x(x_i - \beta y_i + S_i + I_i + \rho) \quad (19a)$$

$$dy_i/dt = -\lambda y_i + f_y(\alpha x_i) \quad (19b)$$

where α and β are coupling parameters between the two units, S_i represents inputs from the other oscillators, and I_i represents external stimulation. Therefore, oscillators communicate through their excitatory units. λ is a decay parameter, and ρ denotes the amplitude of a Gaussian noise term. $f_r(v)$ is a sigmoid function as defined in Equation 4, where $r \in \{x, y\}$. It has been shown that the system defined by Equations 19a and 19b can produce oscillations of different

frequency and amplitude within a wide range of parameters, and the oscillator model can be biologically interpreted as an approximation to a group of excitatory and inhibitory neurons.[37]

The objects to be segmented are presented to the network by simply setting the I_i's of the oscillators stimulated (covered) by the objects to a high value (see Figure 5a). Thus the stimulated oscillators are active, while the others keep silent. The neighborhood connections within the network will then synchronize the oscillators representing a single segment (a connected region). Oscillator groups representing different objects cannot be synchronized because there are no interconnections between them. Figure 5b through 5d display the simulation results for segmenting Cup and Desk Lamp. The phases of the active oscillators are initially randomized. Figure 5b shows a snapshot of the network activity shortly after the network started to evolve according to Equations 19a and 19b. At this time, there was no grouping within each object, and oscillator activities were largely random. Figure 5c shows a snapshot after the system evolved for a short time. We can clearly see the effect of grouping: Oscillators belonging to the same object now have almost the same activity. Also, the activities of the Cup oscillators are much higher than the Lamp oscillators. Figure 5d shows another snapshot, where the Lamp oscillators reach high activity while the Cup oscillators exhibit much less activity. To help illustrate the entire dynamic process, Figure 5e shows the temporal activities of the two oscillators, the upper one representing Lamp and the lower one Cup. An object is synchronized when steady oscillations occur. After several cycles, both objects reach phase synchrony, but Cup reaches it earlier than Lamp, since Cup is smaller and more uniform (Figure 5a). Figure 5e also shows that there is a "chaotic" transient before each object reaches synchrony. I did not try to make the two objects antisynchronized in this simulation, but it is possible to do so with an inhibitory mechanism.

Pattern segmentation is traditionally done by detecting edges and contours, and then labeling different regions based on closed contours.[38,39] The solution provided here operates directly on connected regions, avoiding the detour of detecting contours and labeling regions, which is often an ambiguous process. Using phase synchrony of fast neural oscillations to symbolize object grouping can potentially serve as a mechanism for visual attention, whereby multiple objects can be attended simultaneously. "Simultaneity" is used here in the psychological sense: The basic perception time unit is a multiple of oscillation cycles, during which each object has its chance to reach a peak activity.

Apparently, this segmentation mechanism works regardless of relative positions between images or their shapes. The neighborhood connection pattern in the network can be viewed as a built-in constraint for this task, and no learning is involved. Yet, the solution is indeed generic in the sense that any connected image can be grouped this way. Neighborhood connectivity is important because it preserves the objects' geometrical structure, which is lost in a fully connected network. If we allow lateral connections beyond immediate neighbors, network capability and flexibility will be enhanced markedly. Since lateral connection is a general property of brain organization, it is not hard to imagine that such a style of network might exist in the visual system.

Of course, segmentation of real images is more complicated than just segmenting connected regions. Issues such as object occlusion, intersecting segments, and noise make the problem more difficult. The example provided here illustrates a new approach to tackling the problem of object segmentation by emphasizing emergent properties from local feature similarities, which I believe will play a fundamental role in any successful system of real image segmentation.

INVARIANT PATTERN RECOGNITION will be a problem facing neural networks for some time, and the challenge is to overcome the limitation of Hamming distance generalization. Translation invariance would be easiest to achieve, and there are already networks such as the dynamic link architecture that can handle it quite successfully. Scale and rotation invariance would be harder, but a satisfactory solution can be expected in the near future. The most difficult one is invariance of distortion, which maintains topological structures. Distortion itself contains a variety of transformations. Again, the dynamic link architecture is one of the few networks that directly attack the problem. Proper representation of the object structure will probably be the key to the solution.

In many cases, a visual scene is not recognized as a single pattern. Imagine that when you walk into a classroom, a typical scene is composed of students, a blackboard, desks, and chairs. Before the entire scene is understood, it is first segmented into components (so called preattentive processing), each of which is then recognized, and the entire scene is recognized as a synthesis of its components (in general, segmentation and recognition should be an interactive process in perception). Before such segmentation takes place, no meaningful recognition can be obtained, even with invariant pattern recognition. Pattern segmentation can be based on either input coherency (Gestalt laws) or prior knowledge. The example in Figure 5 demonstrates how segmentation can be accomplished based on features of sensory inputs (connectedness); another neural model[40] has demonstrated segmentation based on stored patterns (prior knowledge). Neural segmentation will be an important topic for future research, and its solution largely depends on how to discover global coherency based on local information and memory, and how to simultaneously represent multiple patterns. Scene analysis can only be addressed after an appropriate segmentation algorithm is in place.

So far, I have only discussed pattern recognition based on spatial features; in addition, time provides another framework for organizing patterns in hearing and vision. Temporal patterns have often been handled as spatial patterns with delay units[41] or temporal decays,[42,43] but these models lack a proper treatment of time. Thus, problems such as speed and distortion invariance remain unsolved. I predict that time will be treated as an independent and major dimension, instead of being secondary to spatial dimensions. Since time is also fundamental for segmentation (for instance, segmentation by motion) and for causal reasoning, temporal information processing will be a major focus in future study.

Neural networks will eventually achieve great success in pattern recognition, and the dynamic link network promises to reap great rewards in this direction. Although the system itself is much less specified than standard ones, it has certainly made progress, particularly by changing the way

the problem is treated. For pattern recognition, generic learning machines will gradually give way to neural networks that encode proper constraints, thus capturing the uniqueness of being a pattern.

Acknowledgment

This work was supported in part by National Science Foundation grant IRI-9211419 and Office of Naval Research grant N00014-93-1-0335. I thank John Kolen for critically reading earlier manuscripts. This paper was submitted April 17, 1992, and accepted April 20, 1993.

References

1. R.P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, Apr. 1987, pp. 4-22.
2. M.A. Arbib, *The Metaphorical Brain 2: Neural Networks and Beyond*, Wiley Interscience, New York, 1989.
3. H. Hertz, A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, Calif., 1991.
4. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.
5. K. Steinbuch, "Die Lernmatrix [The Learning Matrix]," *Kybernetik*, Vol. 1, 1961, pp. 36-45.
6. D.J. Willshaw, O.P. Buneman, and H.C. Longuet-Higgins, "Nonholographic Associative Memory," *Nature*, Vol. 222, 1969, pp. 960-962.
7. J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Nat'l Academy of Science*, Vol. 79, 1982, pp. 2,554-2,558.
8. J.J. Hopfield, "Neurons with Graded Responses Have Collective Computational Properties like Those of Two-state Neurons," *Proc. Nat'l Academy of Science*, Vol. 81, 1982, pp. 3,088-3,092.
9. J.J. Hopfield and D.W. Tank, "'Neural' Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52, 1985, pp. 141-152.
10. D.H. Ackley, G.E. Hinton, and T.J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, Vol. 9, pp. 147-169.
11. J.B. Pollack, "Connectionism: Past, Present, and Future," *Artificial Intelligence Review*, Vol. 3, 1989, pp. 3-20.
12. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, Vol. 1, D.E. Rumelhart and J.L. McClelland, eds., MIT Press, Cambridge, Mass., pp. 318-362.
13. F. Rosenblatt, *Principles of Neural Dynamics*, Spartan, New York, 1962.
14. K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, Vol. 2, 1989, pp. 359-366.
15. B.A. Pearlmutter, "Learning State Space Trajectories in Recurrent Neural Networks," *Neural Computation*, Vol. 1, 1989, pp. 263-269.
16. R.J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, Vol. 1, 1989, pp. 270-280.
17. Y. Le Cun et al., "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, Vol. 1, 1989, pp. 541-551.
18. G.A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphs, and Image Processing*, Vol. 37, 1987, pp. 54-115.
19. G.A. Carpenter and S. Grossberg, "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, Vol. 26, 1987, pp. 4,919-4,930.
20. G.A. Carpenter and S. Grossberg, "ART 3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architecture," *Neural Networks*, Vol. 3, 1990, pp. 129-152.
21. G.A. Carpenter and S. Grossberg, "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network," *Computer*, Mar. 1988, pp. 77-88.
22. C. von der Malsburg, "The Correlation Theory of Brain Function," Tech. Report 81-2, Max Planck Institute for Biophysical Chemistry, Gttingen, Germany, 1981.
23. C. von der Malsburg, "Nervous Structures with Dynamical Links," *Berichte der Bunsengesellschaft für Physikalische Chemie*, Vol. 89, 1985, pp. 703-710.
24. E. Bienenstock and C. von der Malsburg, "A Neural Network for Invariant Pattern Recognition," *Europhysics Letters*, Vol. 4, 1987, pp. 121-126.
25. C. von der Malsburg, "Pattern Recognition by Labeled Graph Matching," *Neural Networks*, Vol. 1, 1988, pp. 141-148.
26. M. Garey and D. Johnson, *Computers and Intractability*, Freeman, New York, 1979.
27. S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, 1983, pp. 671-680.
28. J. Buhmann, M. Lades, and C. von der Malsburg, "Size and Distortion Invariant Object Recognition by Hierarchical Graph Matching," *Proc. Int'l Joint Conf. Neural Networks*, Vol. II, IEEE, Piscataway, N.J., 1990, pp. 411-416.
29. M. Lades et al., "Distortion Invariant Object Recognition in the Dynamic Link Architecture," *IEEE Trans. Computers*, Vol. 42, No. 3, 1993, pp. 300-311.
30. R. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*, Wiley & Sons, New York, 1992.
31. T. Kohonen, "The Self-Organizing Map," *Proc. IEEE*, Vol. 78, 1990, pp. 1,464-1,480.
32. M.A. Arbib, "Schemas and Neural Networks for Sixth-Generation Computing," *Parallel and Distributed Computing*, Vol. 6, 1989, pp. 185-216.
33. D.L. Wang and M.A. Arbib, "How Does the Toad's Visual System Discriminate Different Worm-Like Stimuli?" *Biological Cybernetics*, Vol. 64, 1991, pp. 251-261.
34. S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilemma," *Neural Computation*, Vol. 4, 1992, pp. 1-58.
35. H.J. Reitboeck and J. Altmann, "A Model for Size- and Rotation-Invariant Pattern Processing in the Visual System," *Biological Cybernetics*, Vol. 51, 1984, pp. 113-121.
36. I. Rock and S. Palmer, "The Legacy of Gestalt Psychology," *Scientific American*, Vol. 63, Dec. 1990, pp. 84-90.
37. D.L. Wang, "Emergent Synchrony in Locally Coupled Neural Oscillators," Tech. Report 12/92-TR36, Dept. of Computer and Information Science, Ohio State Univ., Columbus, 1992.
38. D. Geman et al., "Boundary Detection by Constrained Optimization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 12, 1990, pp. 609-628.
39. S. Grossberg and L. Wyse, "A Neural-Network Architecture for Figure-Ground Separation of Connected Scenic Figures," *Neural Networks*, Vol. 4, 1991, pp. 723-742.
40. D.L.Wang, J. Buhmann, and C. von der Malsburg, "Pattern Segmentation in Associative Memory," *Neural Computation*, Vol. 2, 1990, pp. 94-106.
41. R.P. Lippmann, "Review of Neural Networks for Speech Recognition," *Neural Computation*, Vol. 1, 1989, pp. 1-38.
42. D.L. Wang and M.A. Arbib, "Complex Temporal Sequence Learning Based on Short-Term Memory," *Proc. IEEE*, IEEE, Piscataway, N.J., Vol. 78, 1990, pp. 1,536-1,543.
43. R.L. Watrous, "Phoneme Discrimination Using Connectionist Networks," *J. Acoustical Society of America*, Vol. 87, 1990, pp. 1,753-1,772.

DeLiang Wang is an assistant professor in the Dept. of Computer and Information Science and the Center for Cognitive Science at Ohio State University. His research interests include temporal pattern processing, auditory and visual pattern perception, neural mechanisms of visuomotor coordination, neural network theories, and computational neuroscience. He holds a BSc (1983) and a MSc (1986) from Beijing University, Beijing, China, and a PhD (1991) from the University of Southern California, all in computer science.

He can be reached at the Dept. of Computer and Information Science, Ohio State University, Columbus, OH 43210-1277; internet, dwang@cis.ohio-state.edu