Recognition of Spatiotemporal Patterns by Nonmonotone Neural Networks

Masahiko Morita[1] **and Satoshi Murakami**[2]

[1]Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba, Ibaraki 305, Japan
mor@is.tsukuba.ac.jp

[2]Doctoral Program in Engineering, University of Tsukuba, Tsukuba, Ibaraki 305, Japan
mare@bcl.esys.tsukuba.ac.jp

Abstract

A neural network model that recognizes spatiotemporal patterns without expanding them into spatial patterns is presented. This model forms trajectory attractors in the state space of a fully recurrent network by a simple learning algorithm using nonmonotone dynamics. When a spatiotemporal pattern is inputted after learning, the network state is attracted to the corresponding learned trajectory and the incomplete part of the input pattern is restored in the input part of the model; at the same time, the output part indicates which spatiotemporal pattern is being inputted. In addition, this model can recognize the learned patterns correctly even if they are temporally extended or contracted.

1 Introduction

Neural networks for pattern recognition usually deal with a spatiotemporal pattern by expanding it into a spatial pattern using time-delay filters [Tank and Hopfield, 1987] or multilayer delay units [Waibel, 1989]. This method, however, has some drawbacks such that the temporal length of recognizable patterns is limited to the maximum delay time and that temporal extension and contraction is difficult to handle.

Another conventional way to recognize spatiotemporal patterns is to apply an advanced learning algorithm, such as the temporal back-propagation algorithm, to recurrent networks. Those algorithms, however, are quite complicated and require a long time for learning. Besides, they do not work well when the network acts asynchronously or time-continuously, unless the network size is small enough. This is because recurrent networks with conventional type dynamics can hardly make a stable and continuous state transition, since virtually only point attractors which are isolated from each other can be embedded in them [Morita, 1996].

On the other hand, a nonmonotone neural network, namely a recurrent network whose dynamics are modified so that each neuron has nonmonotonic input-output characteristics, not only possesses high memory capabilities [Morita, 1993], but also can stably vary its state along a continuous trajectory which is a string-type attractor. Moreover, such trajectory attractors can be embedded by simple learning based on the covariance rule [Morita, 1996]. Using these properties, we can formulate a new principle of spatiotemporal pattern recognition that does not require synchronization or time-delay mechanisms. Our purpose in this paper is to present a basic model based on such a principle and show its potentiality.

2 Principle

2.1 Structure and Dynamics

This model has a simple structure composed of n neurons with mutual connections. These neurons are divided into two groups, input and output parts, though all neurons obey the same dynamics and learning rule. For convenience, we give serial numbers to the neurons such that neurons 1 to k are the input part and $k+1$ to n are the output part.

Dynamics of the network are expressed by

$$\tau \frac{du_i}{dt} = -u_i + \sum_{j=1}^{n} w_{ij} y_j + z_i, \qquad (1)$$

where u_i is the potential of neuron i, w_{ij} is the synaptic weight from neuron j, z_i is the external input, and τ is a time constant. The output y_i is given by

$$y_i = f(u_i), \qquad (2)$$

where $f(u)$ is a nonmonotonic function as shown in Fig. 1. We use, as the nonmonotonic output function,

$$f(u) = \frac{1-e^{-cu}}{1+e^{-cu}} \cdot \frac{1+\kappa e^{c'(|u|-h)}}{1+e^{c'(|u|-h)}}, \qquad (3)$$

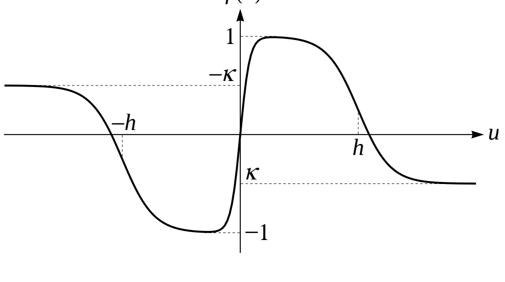

Fig. 1. Nonmonotonic output function.

where c, c', h and κ are constants (we substitute $c = 50$, $c' = 10$, $h = 0.5$, $\kappa = -1$ in the experiments described later).

Since the polarity of u_i is important in nonmonotone neural networks, we consider $x_i = \mathrm{sgn}(u_i)$ and treat the vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ as the network state, where $\mathrm{sgn}(u) = 1$ for $u > 0$ and -1 for $u \leq 0$.

The network state $\boldsymbol{x}$ at an instant is represented by a point in the state space consisting of 2^n possible states. When $\boldsymbol{x}$ changes, it almost always moves to an adjacent point in the state space because x_i changes asynchronously. Consequently, $\boldsymbol{x}$ leaves a track with time, which we call the trajectory of $\boldsymbol{x}$. Similarly, we call $\boldsymbol{x}_{\mathrm{in}} = (x_1, \ldots, x_k)$ and $\boldsymbol{x}_{\mathrm{out}} = (x_{k+1}, \ldots, x_n)$ the states of the input and the output parts, respectively, and consider the trajectories of $\boldsymbol{x}_{\mathrm{in}}$ and $\boldsymbol{x}_{\mathrm{out}}$ in the state space of each part.

2.2 Learning

Let $\boldsymbol{s}^1(t), \ldots, \boldsymbol{s}^m(t)$ be m spatiotemporal patterns to be recognized, where $\boldsymbol{s}^\mu = (s_1^\mu, \ldots, s_k^\mu)$. We assume that the elements s_i^μ are ± 1 and change asynchronously. Then we can consider m trajectories corresponding to $\boldsymbol{s}^\mu$ in the k-dimensional pattern space regarded in the same light as the state space of the input part. These trajectories may intersect or overlap with one another.

We perform learning so that the state $\boldsymbol{x}_{\mathrm{out}}$ of the output part becomes a target state $S^\mu = (s_{k+1}^\mu, \ldots, s_n^\mu)$ when the spatiotemporal pattern $\boldsymbol{s}^\mu$ is inputted into the input part. The learning algorithm is as follows.

First, we create a learning signal vector $\boldsymbol{r} = (r_1, \ldots, r_n)$ with binary elements ($r_i = \pm 1$). The learning signal $\boldsymbol{r}_{\mathrm{in}}$ corresponding to the input part is $\boldsymbol{s}^\mu$, that is, $r_i = s_i^\mu$ for $i \leq k$. The learning signal $\boldsymbol{r}_{\mathrm{out}}$ corresponding to the output part is a spatiotemporal pattern changing gradually from a static pattern O to S^μ, where we assume $O = (-1, \ldots, -1)$ without losing generality. Since $\boldsymbol{r}$ is an n-dimensional binary vector, as well as $\boldsymbol{x}$, $\boldsymbol{r}$ is regarded as moving in the state space of the network from $(\boldsymbol{s}^\mu(0), O) \equiv (s_1^\mu(0), \ldots, s_k^\mu(0), -1, \ldots, -1)$ to $(\boldsymbol{s}^\mu(T), S^\mu)$, where T is the temporal length of $\boldsymbol{s}^\mu$.

Next, we give an initial state $\boldsymbol{x} = (\boldsymbol{s}^\mu(0), O)$ and input $\boldsymbol{r}$ in the form $z_i = \lambda_i r_i$ to the network while it acts according to Equation 1. Here, λ_i denotes the input intensity of r_i, which is a constant λ_{in} for the input part ($i \leq k$) and a variable λ_{out} decreasing with the process of learning for the output part ($i > k$).

We simultaneously modify all synaptic weights w_{ij} according to

$$\tau' \frac{dw_{ij}}{dt} = -w_{ij} + \alpha\, r_i y_j, \tag{4}$$

where τ' denotes a time constant of learning ($\tau' \gg \tau$) and α is a learning coefficient. Since performance of learning is better when α is a decreasing function of $|u_i|$, we put $\alpha = \alpha' x_i y_i$, α' being a positive constant.

When $\boldsymbol{r}$ is moving in the state space, $\boldsymbol{x}$ follows slightly behind, roughly along the same trajectory. When $\boldsymbol{r}$ reaches the end, we keep $\boldsymbol{r} = (\boldsymbol{s}^\mu(T), S^\mu)$ and continue modifying w_{ij} until $\boldsymbol{x}$ comes close enough to $\boldsymbol{r}$.

We apply this procedure to all μ, and repeat it over some cycles, gradually decreasing λ_{out}. If $\boldsymbol{x}_{\mathrm{out}}$ can reach a state near S^μ even when $\lambda_{\mathrm{out}} = 0$, then the learning is completed.

2.3 Recognition

By the above learning, the trajectories of $\boldsymbol{x}$ become attractors of the dynamical system formed by the nonmonotone neural network [Morita, 1996]; that is, there exist m trajectory attractors in the state space after learning. Using this, the spatiotemporal patterns are recognized as follows.

Let us assume that $\boldsymbol{s}' = (s_1', \ldots, s_k')$ is an input pattern made by transforming (e.g., adding noise to) $\boldsymbol{s}^1$ and s_i' is 1, -1 or 0. We input it to the model in the form $z_i = \lambda_{\mathrm{in}} s_i'$ ($i \leq k$). To the output part, we give the initial state O and input nothing ($z_i = 0$) thereafter.

When $\boldsymbol{s}'$ is inputted in this way, $\boldsymbol{x}$ is attracted to the nearest trajectory attractor that is thought to correspond to $\boldsymbol{s}^1$. Consequently, it is expected that the output state $\boldsymbol{x}_{\mathrm{out}}$ becomes nearly equal to S^1 when we finish inputting $\boldsymbol{s}'$.

3 Computer Simulation and Discussion

To confirm the above principle, we carried out computer simulations on the model with 300 input and 200 output neurons ($k = 300$, $n = 500$).

Four spatiotemporal patterns $\boldsymbol{s}^1 = \{ABC\}_{40\tau}$, $\boldsymbol{s}^2 = \{ABD\}_{40\tau}$, $\boldsymbol{s}^3 = \{DAC\}_{40\tau}$ and $\boldsymbol{s}^4 = \{DBC\}_{40\tau}$ were used in the experiment, where A, B, C and D are k-dimensional binary vectors selected at random; $\{ABC\}$ represents the shortest path from A via B to C, and $\{ABC\}_T$ denotes a spatiotemporal pattern whose trajectory is $\{ABC\}$ and temporal length is T. The target states S^1, S^2, S^3 and S^4 were selected at random, but S^1 and S^2 were selected such that they have a similarity of 0.5, where similarity is defined by the direction cosine between two vectors. The reason we make S^1 and S^2 similar is described below.

After finishing 10 cycles of learning, we inputted various patterns and examined the behavior of the model. The parameters were $\alpha' = 2$, $\tau' = 5000\tau$ and $\lambda_{\mathrm{in}} = 0.2$; λ_{out} decreased by degrees from 0.2 to 0.

3.1 Recognition Process

Figure 2 shows a process of recognition when part of $\boldsymbol{s}^1$ was input. Specifically, $z_i = 0.2\, s_i^1$ for half elements of the input part that are randomly chosen and $z_i = 0$ for the other half; at $t > 40\tau$, $z_i = 0$ for all i. Similarities (direction cosines) between $\boldsymbol{x}_{\mathrm{out}}$ and S^μ denoted by $\phi(S^\mu)$ are plotted in the top graph and those between $\boldsymbol{x}_{\mathrm{in}}$ and A to D are in the bottom one. The abscissa is time scaled by the time constant τ.

The similarity $\phi(A)$ between $\boldsymbol{x}_{\mathrm{in}}$ and A increases very rapidly from the initial value 0.5 to more than 0.9 and

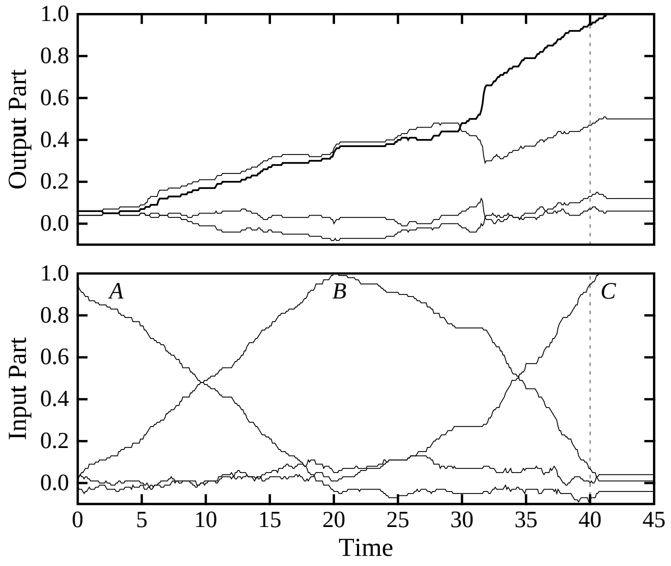

Fig. 2. A process of recognition. Spatiotemporal pattern $\boldsymbol{s}^1$ is inputted with 50% elements blank.

then decreases gradually. In this process, $\phi(A) + \phi(B)$ is constant and is nearly equal to 1, which indicates that $\boldsymbol{x}_{\text{in}}$ is moving along the path $\{AB\}$. We also see that $\boldsymbol{x}_{\text{in}} \simeq B$ at $t = 20\tau$ and $\boldsymbol{x}_{\text{in}} \simeq C$ at $t > 40\tau$ and that the whole of $\boldsymbol{s}^1$ is restored in the input part.

On the other hand, the similarity $\phi(S^1)$ between $\boldsymbol{x}_{\text{out}}$ and S^1 (thick line) increases consistently with time and $\boldsymbol{x}_{\text{out}} \simeq S^1$ at $t > 40\tau$. This means that the model has correctly recognized the input pattern as $\boldsymbol{s}^1$. We should note that the trajectory $\{ABC\}$ of $\boldsymbol{s}^1$ overlaps everywhere with other trajectories and thus $\boldsymbol{s}^1$ cannot be distinguished by the instantaneous input pattern at any moment.

We should also note that $\phi(S^1)$ and $\phi(S^2)$ rise in the same manner while $t < 25\tau$ and rapidly separate at $t \simeq 30\tau$. This indicates that $\boldsymbol{x}_{\text{out}}$ moves in the middle of trajectories $\{OS^1\}$ and $\{OS^2\}$ at first and then approaches $\{OS^1\}$ when $\boldsymbol{x}_{\text{in}}$ approaches C after passing through B.

This process is schematically shown in Fig. 3, where the n-dimensional state space of the network is represented three-dimensionally. Two figures (a) and (b) depict the same thing from different angles. The origin represents the initial state $\boldsymbol{x} = (A, O)$ meaning $\boldsymbol{x}_{\text{in}} = A$ and $\boldsymbol{x}_{\text{out}} = O$. The thick line represents the trajectory of $\boldsymbol{x}$ and the broken lines represent the trajectories $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ of the learning signal for $\boldsymbol{s}^1$ and $\boldsymbol{s}^2$. The gray lines represent the projection onto the x_1–x_2 or x_3–x_4 plane, that is, the trajectories in the state space of the input or output part.

If we observe only the input part, the two trajectories $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ overlap in their first half and then diverge in their second. On the other hand, the trajectories in the output part diverge at the starting point. Thus as a whole, $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ are separate but rather close in the first half.

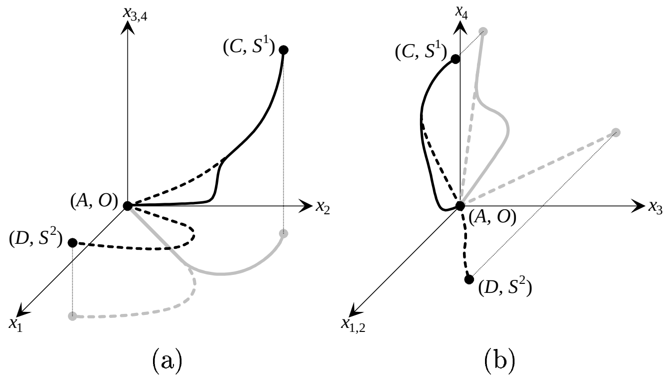

Fig. 3. Schematic of the recognition process. (a) x_1–x_2 plane represents the input part, and $x_{3,4}$ axis the output part; (b) $x_{1,2}$ represents the input part, and x_3–x_4 the output part.

The nonmonotone neural networks have a property that when some attractors exist nearby, states lying between them are comparatively stable [Morita, 1996]. In other words, the energy landscape of the network has a "flat" bottom between neighboring attractors because attractive force is smaller near attractors (note that in the case of conventional neural networks, the energy landscape is "sharp" at attractors). Consequently, while $\boldsymbol{x}_{\text{in}}$ is moving along the common path $\{AB\}$, $\boldsymbol{x}_{\text{out}}$ moves in the middle of $\{OS^1\}$ and $\{OS^2\}$.

As $\boldsymbol{x}_{\text{in}}$ departs from B toward C, the distance from $\boldsymbol{x}$ to $\boldsymbol{r}^2$ increases rapidly whereas that to $\boldsymbol{r}^1$ does not increase to such an extent, so that $\boldsymbol{x}$ cannot remain in between the two. As a result, $\boldsymbol{x}$ is attracted to $\boldsymbol{r}^1$ and $\boldsymbol{x}_{\text{out}}$ approaches S^1.

We can see from the above discussion why a spatiotemporal pattern $\{OS^\mu\}_T$, rather than a static pattern S^μ, should be used for the learning signal $\boldsymbol{r}_{\text{out}}$. That is, if $\boldsymbol{r}_{\text{out}}$ is a static pattern, $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ are far apart over the entire path and thus $\boldsymbol{x}$ is attracted to either of the two soon after starting from the origin; once $\boldsymbol{x}$ is attracted to $\boldsymbol{r}^1$, for example, $\boldsymbol{x}$ can hardly transfer to $\boldsymbol{r}^2$ even if $\boldsymbol{x}_{\text{in}}$ goes along $\{BD\}$ afterwards.

From similar reasoning, if $\boldsymbol{s}^1$ and $\boldsymbol{s}^2$ are identical (or very similar) in their first part, the corresponding target states S^1 and S^2 should be similar so that the distance between $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ is decreased. Then there is a less possibility that $\boldsymbol{x}$ will be attracted to $\boldsymbol{r}^\mu$ before sufficient information is inputted into the model, and even if it occurs, $\boldsymbol{x}$ can transfer to the correct trajectory more easily.

3.2 Recognizing Patterns with Blank Sections

The trajectory attractor formed by the above learning not only has a strong flow surrounding it that runs into it, but also has a gentle flow that moves as fast as $\boldsymbol{r}$ along the trajectory [Morita, 1996]. Accordingly, this model can recognize a learned spatiotemporal pattern even if the input pattern has some blank sections.

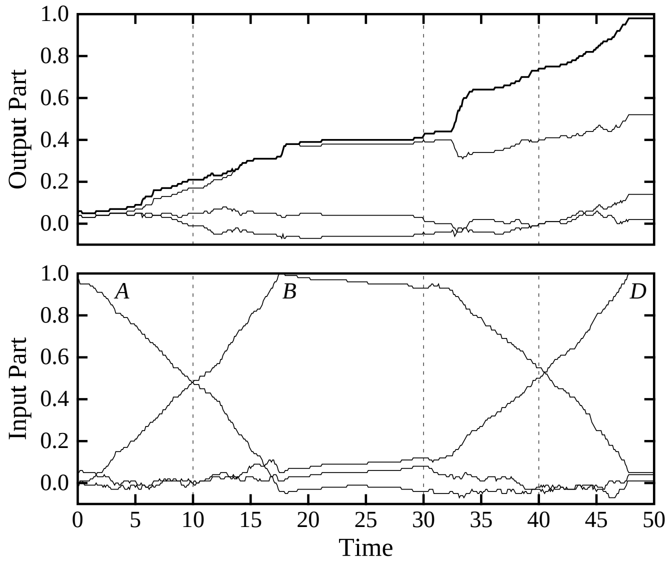

Fig. 4. Behavior of the model when some temporal sections of the input pattern are omitted. No external input is given while $10\tau < t < 30\tau$ and $t > 40\tau$.

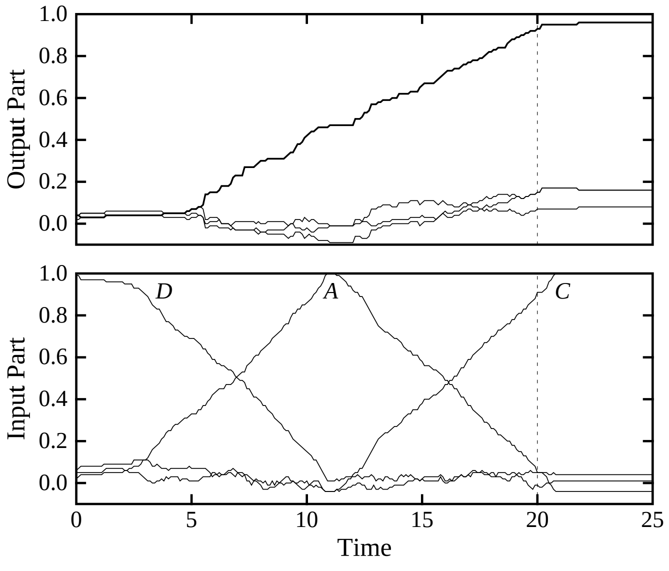

Fig. 5. Behavior when a learned pattern is inputted at a different pace. The temporal length T of the input pattern is contracted by half ($T = 20\tau$).

As an example, we inputted the first quarter of $\boldsymbol{s}^2$ (from A to the midpoint between A and B) for $0 \leq t \leq 10\tau$ and then cut off the input. Figure 4 shows the behavior of the model in the same way as shown in Fig. 2, but the thick line represents $\phi(S^2)$. We see that the movement of $\boldsymbol{x}$ is roughly the same as that in Fig. 2 for $10\tau < t \leq 20\tau$ although there is no external input. However, when $\boldsymbol{x}_{\text{in}}$ passed through B and slightly approached C and D, $\boldsymbol{x}$ stopped. This is thought to be an equilibrium state in which the attracting forces from multiple attractors balance out.

Then we inputted the third quarter of $\boldsymbol{s}^2$ (from B to the midpoint between B and D) for $30\tau \leq t \leq 40\tau$, and cut off the input again. We see that $\boldsymbol{x}$ is attracted to $\boldsymbol{r}^2$ and finally comes close to (D, S^2).

In this way, this model complements the blank sections of the input pattern and recognizes it correctly, provided that no other trajectory attractors exist near the blank sections.

3.3 Recognizing Patterns with Temporal Extension and Contraction

As described above, $\boldsymbol{x}$ basically moves at the same pace as that in learning after being attracted to the learned trajectory. However, if $\boldsymbol{s}^\mu$ is inputted at a different pace, or $\boldsymbol{s}'$ is a temporally extended or contracted pattern of $\boldsymbol{s}^\mu$, $\boldsymbol{x}_{\text{in}}$ follows the input pattern unless the pace is too fast. Also, $\boldsymbol{x}_{\text{out}}$ keeps pace with $\boldsymbol{x}_{\text{in}}$ and approaches S^μ.

Figure 5 shows the recognition process when $\boldsymbol{s}^3$ was inputted at double the pace, or $\boldsymbol{s}' = \{DAC\}_{20\tau}$. We see that the input pattern is correctly recognized, though the transition of $\boldsymbol{x}_{\text{in}}$ is slightly delayed. In the same way, this model can recognize temporally extended patterns.

4 Concluding Remarks

We have described a model which can recognize spatiotemporal patterns by the use of trajectory attractors formed in a nonmonotone neural network. This model not only contains many interesting features, but also seems much more similar to the brain in its working principle than the conventional models of spatiotemporal pattern recognition.

The model described in this paper is a basic one and there is much room for further development. For example, to recognize more complex patterns, we can introduce a middle part or hidden neurons between the input and output parts [Morita and Murakami, in preparation]. Theoretical analysis and application to speech recognition are also subjects for future study.

Acknowledgment

This study was supported by a Grant-in-Aid for Scientific Research (#08279105, #08780328 and #0978031) from the Ministry of Education of Japan.

References

[Morita, 1993] M. Morita, Associative memory with nonmonotone dynamics, *Neural Networks* **6,** 115–126, 1993.

[Morita, 1996] M. Morita, Memory and learning of sequential patterns by nonmonotone neural networks, *Neural Networks* **9,** 1477–1489, 1996.

[Tank and Hopfield, 1987] D. W. Tank and J. J. Hopfield, Neural computation by concentrating information in time, *Proc. Natl. Acad. Sci. USA* **84,** 1896–1900, 1987.

[Waibel, 1989] A. Waibel, Modular construction of time-delay networks for speech recognition, *Neural Computation* **1,** 328–339, 1989.