Learning and Retrieving Spatio-Temporal Sequences with Any Static Associative Neural Network

Lipo Wang

Abstract—The purpose of the present paper is to report on the design of a general system that is capable of learning and retrieving spatio-temporal sequences using any static associative neural networks (ANN’s), including both autoassociative and heteroassociative neural networks. This artificial neural system has three major components: a voting network, a parallel array of ANN’s, and delayed feedback lines from the output of the system to the ANN layer. The system has separate primary and pairing input channels. During learning, pairs of sequences of spatial patterns are presented to the primary and the pairing input channels simultaneously. During retrieving, a cue sequence, which is presented to the primary input channel only, and which may have spatial imperfections and temporal gaps, causes the system to output the stored spatio-temporal sequence. As a demonstration of the applicability of the present general system, we present an implementation using a specific neural network, that is, our dynamically generated variant of the counterpropagation network. This system shows computational advantages such as fast and accurate learning and retrieving, and the ability to store a large number of sequences consisting of nonorthogonal spatial patterns.

Index Terms—Autoassociative, heteroassociative, neural network, noise, spatio-temporal sequence.

I. INTRODUCTION

R ECENTLY there has been extensive research activity in learning and generating temporal phenomena with associative neural network (ANN) models. The main motivation for creating and studying these models is to make both systematic investigations and analytic statements, and to identify the main attributes of the most important mechanisms of information storage and processing in biological neural networks, thereby creating artificial systems that mimic or even surpass certain aspects of biological intelligence. Temporal phenomena in ANN’s are particularly interesting because we ourselves, together with the machines we build, function in a dynamic world.

Grossberg presented his Avalanche model in the form of delayed partial–differential–difference equations [12]. He proved mathematically that the Avalanche model is capable of learning spatio-temporal sequences after an infinite number of presentations of the training sequences; however, the model is computationally expensive, and its practical capabilities to effectively generate spatio-temporal sequences, such as the ones discussed in the present paper, are yet to be demonstrated.

Fukushima [11] invented a temporal system consisting of a number of McCulloch–Pitts neurons that are fully connected with Hebb-type [15] dot-product synapses. Any two neurons are connected with multiple synapses with different time delays. This system showed limited ability to learn spatio-temporal sequences, that is, it required many iterations for sequence retrieval and retrieved nonorthogonal patterns with difficulty. Images generated by this system are often obscured by noise (spurious memories).

Kosko [20] proposed the bidirectional associative memory that consists of interconnected networks, and is able to produce spatio-temporal sequences. Bradski et al. [1]–[3] coupled two ART networks [6] to form sequence-producing systems (also see [14]). The capabilities of these networks in handling complicated sequences have not been clearly substantiated with examples such as the ones given in the present paper.

Wang and Yuwono [35] have recently proposed a novel network to generate sequences [33], [34]. This network, like many other sequence-processing systems, does not accept spatio-temporal sequences as input or generate spatio-temporal sequences directly; rather, it learns and produces sequences consisting of components represented by scalar values or symbols. The work of Wang and Yuwono [35] bears some resemblance to the concurrent effort of Nigrin [23]. These networks memorize sequences with the competitive learning algorithm used in the ART networks, and anticipate upcoming components in a sequence with feedback connections. They use decreasing activation in neurons to represent order information in a sequence. Nigrin pointed out that undesirable
interference can occur in his network during learning, which leads to errors in sequence classification. Nigrin’s network classifies sequences (also see [6]), rather than generating sequences—as Wang and Yuwono’s network does [35]. In the present paper, we restrict our discussions to sequence storage and retrieval only; extending this work to sequence classification and discrimination will be the subject of future studies.

Time delays have been used in Hopfield networks [17], [18] to generate spatio-temporal sequences [19], [27], [28] and to process speech signals [31]. These systems also use Hebb-type learning rules, and have rather low memory capacity. Guyon et al. [13] discussed a temporal system that required a priori analytical expressions of all stored sequences. Time delays have also been incorporated into backpropagation networks [26] to form recurrent networks [30] in processing temporal speech signals [21], [32], although backpropagation networks have long learning times. Buhmann and Schulten [4] used stochastic noise to induce transitions between spatial patterns in Hopfield networks, and these transitions formed spatio-temporal sequences. Other existing mechanisms for temporal sequence generation are time-dependent [9], [25], asymmetric [8], [24], and diluted higher order [36], [37], [40], [41] synaptic interactions.

These existing networks for sequence generation are all based on some specific learning rules and network architecture. The purpose of the present study is to provide a general framework for learning and generating spatio-temporal sequences using any static associative ANN’s. This paper is organized as follows. We describe the architectural design of the present system in Section II. The system operations at the learning stage and the retrieving stage are presented in Sections III and IV, respectively. As a demonstration of the applicability of the present general system, we present in Section V an implementation using a specific neural network, that is, our dynamically generated variant of the counterpropagation network, together with a specific voting network based on the variant counterpropagation network. We present further analysis and some limitations of the system in Section VI. Finally, we summarize the results in Section VII.

II. SYSTEM ARCHITECTURE

Spatio-temporal sequences in the present paper denote time-dependent sequences in which each frame or element at any given time is a spatial pattern. Fig. 1 shows three examples of such sequences, which we have created to demonstrate the functionality of the present system.

Sequence (a) ABCDEFGHIABCDE⋯, Sequence (b) 12345612345⋯, Sequence (c) JKLJMNJOPJKLJMJ⋯

Note that in Fig. 1, pattern I, which is the same as pattern 1, appears in both sequences (a) and (b), and pattern J appears more than once within sequence (c). We use boldface to denote spatial patterns throughout this paper. The cyclic nature of the sequences is not required in order to use our model. In the present paper, we are interested in a system for spatio-temporal sequence generation that is able to memorize these sequences after some presentations of these sequences or some variations of these sequences. After learning, the system should be able to recall an entire sequence after being presented with only a small portion of this sequence which may also have spatial imperfections and/or temporal gaps.

Fig. 2 shows the architectural design of our general system for spatio-temporal sequence generation capable of achieving the above objective. The main idea is to use a delayed sequence of heteroassociators to “vote” on the next output at each time step. The detailed operations of the system will be described in subsequent sections.

The system has three major components: a voting network, a parallel array of \( N_L \) heteroassociative neural networks (HANN’s), and time delays that feed the overall output of the system back to the neural network layer. There are \( N \) output
neurons and \( N \) input neurons in each HANN (see Fig. 3 and Section V for the description of a specific HANN), as well as in the voting network (see Fig. 4 and Section V for the description of a specific design of the voting network). The time delay leading to the \( l \)th HANN delays the signal by \( l \) time steps with respect to the current time, where \( l = 1, 2, \ldots, N_L \). The \( i \)th output neuron of each HANN is connected to the \( i \)th input neuron of the voting network, and the \( i \)th output neuron of the voting network is connected to the \( i \)th input neuron of each HANN through the delayed feedback lines. The output neurons of the voting network provide the overall output of the system. The system has two separate input channels: the \textit{primary} and the \textit{pairing} input channels. The \textit{primary} input channel is connected to the input neurons of each HANN through the time delays, whereas the \textit{pairing} input channel is connected to the output neurons of each HANN directly.

A HANN is a neural network that associates a spatial pattern \( P_1 \) with another pattern \( P_2 \) (which may or may not be the same as pattern \( P_2 \)), whereas an autoassociative neural network (AANN) associates a spatial pattern with itself, i.e., \( P_1 = P_2 \) in an AANN. For example, the backpropagation network [26], the counterpropagation network [16], and the bidirectional associative memory (BAM) [20] are HANN’s, whereas the Hopfield network [17] is an AANN. We will describe a specific HANN in detail in Section V.

We assume that there are two separate input channels in each HANN, i.e., the \textit{primary} and the \textit{pairing} input channels. During learning, pairs of \textit{primary} and \textit{pairing} input patterns are presented to a HANN through its \textit{primary} input channel and \textit{pairing} input channel, respectively, and the HANN learns the association between the \textit{primary} and the \textit{pairing} patterns in each pair. After learning and during retrieving, the HANN outputs the corresponding \textit{pairing} pattern when presented with a \textit{primary} pattern which may have spatial imperfections, such as random noise and differences in size, location, and orientation.

The voting network selects and outputs the most popular pattern from the outputs of all embedded HANN’s during retrieving. We will discuss the operation of the voting network further in Section IV, and a specific design of the voting network will be presented in Section V.

III. LEARNING

The operations of the present spatio-temporal system consist of two stages: the learning stage, during which the system learns spatio-temporal sequences; and the retrieving stage, during which the system retrieves sequences after being presented with cues.

At the learning stage, pairs of sequences of spatial patterns are presented to the \textit{primary} and the \textit{pairing} input channels of the system simultaneously, one spatial pattern at each time step.

Whenever there is an external input to the \textit{primary} input channel of the system, we let the system simply output these external signals, regardless of the outputs from the voting network. It is equivalent to disable the voting network whenever the \textit{primary} input channel of the system is used. Since there are always \textit{primary} inputs to the system during learning, the voting network stays disabled at the learning stage.

One of the two sequences in each training pair, the \textit{primary} sequence, is fed into the \textit{primary} input channel, and is then directed into each embedded HANN through appropriate delayed feedback (Fig. 2). The \textit{pairing} sequence in the pair, the expected output of the system corresponding to the \textit{primary} sequence, is fed into the \textit{pairing} input channel, and is then directed into each embedded HANN without delays (Fig. 2).

As shown more clearly below, learning of the spatio-temporal sequences is achieved with the heteroassociative memories in these HANN’s. Thus, the memory capacity of the system will depend on that of the HANN’s used in each specific implementation of the general framework.

The learning process can be best demonstrated with the actual spatio-temporal sequences shown in Fig. 1. Let us consider three HANN’s in the neural network layer (\( N_L = 3 \).
TABLE I

<table>
<thead>
<tr>
<th>t</th>
<th>SYSTEM Primary</th>
<th>HANN 1 Primary</th>
<th>HANN 1 Pairing</th>
<th>HANN 2 Primary</th>
<th>HANN 2 Pairing</th>
<th>HANN 3 Primary</th>
<th>HANN 3 Pairing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>5</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>7</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>8</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>9</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
</tbody>
</table>

For instance, sequence (a) in Fig. 1(a) is presented simultaneously to the primary input and the pairing input channels of the system during learning. Table I shows the primary inputs and the pairing inputs for the overall system and the 3 HANN's during training in the case of Fig. 1(a).

TABLE II

<table>
<thead>
<tr>
<th>t</th>
<th>SYSTEM Primary</th>
<th>HANN 1 Primary</th>
<th>HANN 1 Pairing</th>
<th>HANN 2 Primary</th>
<th>HANN 2 Pairing</th>
<th>HANN 3 Primary</th>
<th>HANN 3 Pairing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

The processes in which the system learns sequences (a) and (c) are recorded in Tables I and II for references in the next section.

IV. RETRIEVING

At the retrieving stage after learning, a cue sequence is presented to the system through the primary input channel only (see Fig. 2). The pairing input channel of the system is not used during retrieving. During the presentation of a cue sequence, the voting network is disabled, and the system simply outputs the cue sequence; the voting network is enabled at the conclusion of the cue presentation. A “plurality vote” is carried out among the outputs of all embedded HANN’s at each subsequent time step.

As we shall see more clearly with examples below, if an unknown or ambiguous sequence is presented to the system, the HANN’s in the system may output “don’t know” or conflicting answers. If the most popular output from the HANN’s is the “don’t know” answer, the voting network outputs an overall “don’t know” answer, and halts further operation. If a plurality cannot be achieved—for example, the voting leads to a tie among conflicting answers—the overall system output also becomes a “don’t know.” Otherwise, the output of the voting network is the most popular pattern from all outputs of the HANN’s that carry signals. Like the “unclassified” result in a classification system, the “don’t know” answer in the present system reduces the chance of outputting a meaningless or erroneous sequence, thereby reducing the error rate. There may be various ways to implement such a voting algorithm; in Section V, we will present a specific neural network design for such a voting algorithm based on the counterpropagation network discussed in that section.

Let us assume that after having learned the association between patterns P1 and P2, and when presented with pattern P1 (a noisy version of pattern P1), the HANN outputs either pattern P2 or a “don’t know” answer. We use the examples shown in Fig. 5 to illustrate the system operation at the retrieving stage.

In Fig. 5(a), a noisy pattern A is presented to the system’s primary input channel at time t = 1, so the system outputs pattern A at time t = 1. At t = 2, the primary input for the first HANN is the output of the system 1 time step before, which is pattern A. Since the pairing input pattern corresponding to pattern A during learning is pattern B (Table I), HANN 1 outputs pattern B at t = 2. At time t = 2, other HANN’s do not respond since they did not receive any input. Hence, the output of the system at time t = 2 is pattern B... The system thus outputs sequence (a) when presented with a noisy pattern...
Fig. 5. Operations of the system during retrieving. (a) Retrieval of sequence Fig. 1(a) with a single spatial pattern as the cue. (b) Response to an ambiguous cue. (c) Unambiguous retrieval of sequence Fig. 1(b). (d) Response to a nonorthogonal cue pattern. (e) Response to an unknown cue sequence. (f) Response to a cue sequence containing temporal gaps denoted by \( \square \). (g) Retrieval of sequence Fig. 1(c).

**TABLE IV**

<table>
<thead>
<tr>
<th>t</th>
<th>SYSTEM Input</th>
<th>HANN 1 Input</th>
<th>HANN 1 Output</th>
<th>HANN 2 Input</th>
<th>HANN 2 Output</th>
<th>HANN 3 Input</th>
<th>HANN 3 Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>A</td>
<td></td>
<td>B</td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>A</td>
<td></td>
<td>C</td>
<td>A</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>B</td>
<td></td>
<td>D</td>
<td>B</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

(A (Table IV summarizes the primary input and the output of the overall system, as well as those of each HANN, at each time step).

In Fig. 5(b), pattern 1 (note that pattern 1 is the same as pattern 2) is presented to the system at time \( t = 1 \); HANN 1 outputs a mixture of patterns 2 and A, i.e., \( (2 + A) \), since HANN 1 has associated pattern 1 with both patterns 2 and A during learning (see Tables I and II). The system also outputs \( (2 + A) \) at \( t = 2 \). At \( t = 3 \), the primary input to HANN 1 is \( (2 + A) \), which prompts HANN 1 to output a “don’t know” answer since it has not learned to associate pattern \( (2 + A) \) with any pattern during learning (Tables I and II). At the same time \( t = 3 \), the primary input is 1, and the output of HANN 2 is \( (3 + B) \) (Tables I and II). The voting network or the overall system outputs a “don’t know” since there is no tie between the outputs of the two signal-carrying HANN’s, with one vote for “don’t know” and one vote for \( (3 + B) \) (Table V).

In Fig. 5(c), a longer piece of the sequence, e.g., 561, is presented to the system instead of pattern 1 alone, the system is able to use the cue to retrieve sequence (Table VI). Likewise, the retrieval will be sequence (a) if 5671 is presented to the system instead of pattern 1 alone.

The system operation in Fig. 5(d) is similar to that in Fig. 5(a) (Table VII). In Fig. 5(e), a sequence \( (DCB) \) unknown to the system is presented. The voting network halts the feedback operation, and outputs a “don’t know” answer since the three HANN’s output conflicting answers: A, E, and G (see the associations in Table I), respectively (Table VIII).

Fig. 5(f) and Table IX show how the system responds to temporal gaps in cue sequences, as occurs when two frames of the...
The primary inputs and the outputs of the overall system and the 3 HANN’s during testing in the case of Fig. 5(f), where temporal gaps (denoted by □) exist in the cue sequence.

<table>
<thead>
<tr>
<th>t</th>
<th>SYSTEM</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
| ... | ... | ... | ... | ...

The primary inputs and the outputs of the overall system and the 3 HANN’s during testing in the case of Fig. 5(g).

<table>
<thead>
<tr>
<th>t</th>
<th>SYSTEM</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>J</td>
<td>J</td>
<td>J</td>
<td>J</td>
</tr>
<tr>
<td>4</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>5</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>6</td>
<td>J</td>
<td>J</td>
<td>J</td>
<td>J</td>
</tr>
</tbody>
</table>
| ... | ... | ... | ... | ...

Note that \( \bar{v}_i^{\text{new}} \) is not the weight vector of the \( i \)th output neuron. Thus, after training, the incoming weights of each neuron in the competitive layer represent a cluster in the primary training patterns, and the outgoing weights connected to this neuron represent the pairing training pattern. If the cluster represented by the incoming weights of a competitive neuron has been associated with multiple pairing patterns during training, the outgoing weights connected to this competitive neuron are an average or mixture of these multiple pairing patterns, as indicated by (2). For example, suppose, at one time, competitive neuron \( k \) wins the competition when the network is presented with a primary input pattern 1 and a pairing input pattern 2; then the incoming weights of this neuron are modified to represent pattern 1, and the outgoing weights are modified to represent pattern 2. Suppose, at another time, the network is presented again with a primary input pattern 1, but together with an pairing input pattern \( \mathbf{A} \); then the incoming weights of this neuron still represent pattern 1; however, the outgoing weights connected to this neuron become a mixture of pattern 2 and pattern \( \mathbf{A} \), according to (2). Because of the resemblance with Grossberg’s Outstar [12], the output layer of the FOCPN is also called the Grossberg layer by Hecht-Nielsen [16].

When a test pattern \( \bar{x}_k \) is presented to the FOCPN after training, the neuron in the competitive layer with weights most similar to the test pattern wins the competition, and broadcasts the associated pairing pattern \( \bar{y}_k \) to the output neurons.

B. Variant of the Forward-Only Counterpropagation Network

In our variant of the FOCPN, the competitive layer and all weights of the network are dynamically generated according to the following competitive learning algorithm used in the ART network [5]. This dynamically generated FOCPN not only achieves desired training and retrieving objectives, but is also efficient to implement in software [29], and has virtually unlimited storage capacity. In addition, it is able to output a “don’t know” answer, which helps to reduce the possibility of making an error. All neurons in the network are assumed to be standard McCulloch–Pitts binary neurons [22]; however, they can be analog neurons with continuously valued outputs as in the original FOCPN [16].

Before training, the network has \( N_i \) input neurons and \( N_o \) output neurons, where \( N_i \) and \( N_o \) are the dimensions of the primary and pairing patterns, respectively, but has
no competitive neurons or weights. When the first pair of primary and pairing training patterns \((\vec{x}_1, \vec{y}_1)\) is presented to the network, the first competitive neuron is generated, with its incoming weights being the primary training pattern \((\vec{u}_1 = \vec{x}_1)\), and the outgoing weights connecting this neuron to all output neurons being the pairing training pattern \((\vec{u}_1 = \vec{y}_1)\).

For each subsequent pair of primary–pairing association, a competition is carried out in the competitive layer, and the neuron whose weights are the most similar to the primary training pattern is over a vigilance threshold, which indicates that the winning neuron is generated, so as to represent a new cluster. We let [42]

\[
\alpha(t) = \alpha'(t) = 1/\tau_i
\]

where \(\tau_i\) is the number of times that the weights of competitive neuron \(i\) have been updated (\(\tau_i = 1\) when neuron \(i\) is first created), so that the incoming and outgoing weights connected to neuron \(i\) are exactly the overall averages of the primary and pairing training patterns used to modify the weights of this neuron:

\[
\vec{u}_i(\tau_i) = \frac{1}{\tau_i} \sum_{\tau' = 1}^{\tau_i} \vec{x}(\tau')
\]

and

\[
\vec{u}_i(\tau_i) = \frac{1}{\tau_i} \sum_{\tau' = 1}^{\tau_i} \vec{y}(\tau'),
\]

After training and during retrieving, test patterns are input to the network from the primary input channel. The pairing input channel is not used during retrieving. For each test pattern, a competition is carried out, and the competitive neuron whose incoming weights have the largest similarity to the test pattern wins the competition. If this largest similarity is above the vigilance threshold, indicating that the test pattern belongs to a learned cluster, the winner neuron broadcasts the associated pairing pattern(s) represented by its outgoing weights to the output neurons of the network. If this largest similarity is below the vigilance threshold, indicating that the test pattern does not belong to any learned cluster, the network outputs a “don’t know” answer.

The choices of the similarity measure and the vigilance threshold are rather arbitrary. For a similarity measure, one may use Hamming distance, Euclidean distance, Manhattan distance, or dot product (inner product) [7]. One may or may not include normalization of the patterns or weights when using these similarity measures, and the normalization may be done according to either the Euclidean norm or the Manhattan norm. For example, if the patterns or weights are normalized according to the Euclidean norm, the dot product becomes the directional cosine [7].

The choice of vigilance threshold may affect both learning and retrieving. Too high a vigilance threshold may result in too many similar clusters or competitive neurons and their weights during learning, and a demand for too much storage space. If the vigilance threshold is too low during learning, one risks clustering dissimilar primary training patterns and mixing their associated pairing patterns. During retrieving, too high a vigilance threshold leads to inadequate tolerance over imperfection in test patterns, and can therefore lead to too many “don’t know” answers. However, if the vigilance threshold is too low during retrieving, the error rate can become too high to be acceptable. Hence, the vigilance threshold, which can be the same or different during learning and retrieving, should be selected according to the acceptable numbers of “don’t know” answers and errors for each set of training and test patterns. A more general discussion of the choices of similarity measure and vigilance threshold is out of the scope of the present work.

C. Design for the Voting Network

We now present a network design that facilitates the voting required in the system, based on our variant of the FOCPN. Before votes can be counted, the output patterns from the HANN’s need to be grouped into distinct categories or clusters. Thus, we may use an architecture similar to that of the variant FOCPN discussed in the previous subsection by letting the pairing input channel coincide with the primary input channel (see Fig. 4).

For the clusters to form, this voting network accepts the output patterns from the HANN’s asynchronously or serially. After learning within the voting network in the same way as described in the previous subsection, the incoming weights (which are identical to the outgoing weights) of each competitive neuron in the voting network represent a voting category. We assume that after the voting network finishes grouping all output patterns from the HANN’s, the most popular category, which is represented by the competitive neuron in the voting network with the highest number of updates \(\tau_{\text{max}}\), wins the final competition, and broadcasts the (average) pattern in this category as the output of the voting network, which is also the overall output of the system. If there is more than one competitive neuron in the voting network with the same value for \(\tau_{\text{max}}\), or if the most popular category is the “don’t know” answer, then the voting network (and thus the overall system) outputs a “don’t know” answer.

We assume that there are no overall system outputs during the above voting process, and only the final outcome of the voting process becomes the overall system output. The voting network is “reset” (all competitive neurons in the voting network are eliminated so that the voting network is ready for processing) before each voting starts.

In the next subsection, we will use this design of the voting network, together with the variant FOCPN as the HANN, in a specific implementation of the general system. However, this particular voting network may be used in conjunction with other HANN’s.

D. Specific Implementation of the General System

We set dimensions of both the primary and pairing spatial patterns, as well as the numbers of neurons in the input and
VI. ANALYSIS AND LIMITATIONS

When dealing with simple situations, in which each spatial pattern in the stored sequences can be uniquely determined by the immediately preceding spatial pattern in the sequence, we can use only one HANN in the present system ($N_L = 1$). For example, if we need a system to learn only sequence (a) and not sequences (b) or (c) in Fig. 1, one HANN will be sufficient since pattern $A$ uniquely determines pattern $B$. Pattern $B$ uniquely determines pattern $C$, and so on.

From the discussions in the earlier sections, we see that it is essential to use more than one HANN and time delay in the present system when at least one spatial pattern in the stored sequences needs to be determined by the spatial patterns that appear earlier than the immediately preceding pattern, such as the three sequences given in Fig. 1. In this section, we discuss how the number of HANN's needed ($N_L$) depends on the complexity of the training sequences; we also discuss some limitations of the system.

To ensure successful retrieving of a spatial–temporal sequence with the present system, the correct spatial pattern must win a plurality vote at each time step. If a spatial pattern appears more than once in the training sequences, we call it a repeating pattern. A repeating pattern may be associated with more than one other spatial pattern, and may therefore lead to mixtures of training patterns, which become unknown to the HANN’s and cause “don’t know” or conflicting answers. In contrast, a nonrepeating pattern is only associated with one pattern during learning, and thus causes unambiguous output during retrieving. To guarantee that this unambiguous output pattern wins in voting, the inputs to the HANN’s must consist of more nonrepeating patterns than repeating patterns at each time step during learning. Hence, the task of finding an appropriate number of HANN’s for a system to learn a given set of spatial–temporal sequences reduces to finding the smallest integer $N_L$ such that there are fewer than $N_L / 2$ repeating patterns in all subsequences of length $N_L$ in all training sequences, where a subsequence of length $g$ in sequence $P_1P_2\cdots P_m$ is $P_iP_{i+1}\cdots P_{i+g-1}$, and $i = 1, 2, \ldots, m - g + 1$.

For example, pattern $I$ (being the same as pattern $1$) appears twice in the sequences shown in Fig. 1, and is associated with multiple spatial patterns during learning (see Table I and II), which causes a “don’t know” answer during retrieval (see Table V). Pattern $J$ also appears more than once. We have chosen $N_L = 3$, which assures that less than half of the patterns in each subsequence of length 3 in all training sequences are repeating patterns ($1 < 3/2 = 1.5$), and thus leads to successful retrievals. If we need a system to learn sequence $P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}$, in which $P_5$ and $P_6$ appear more than once, we should use five HANN’s. The verification of learning and retrieving this sequence with a system of five HANN’s can be carried out in the same fashion as shown in Tables I–X and is omitted here.

The above analysis indicates a limitation of the present system: if such an $N_L$ does not exist for a given set of training sequences, the present system is not able to learn and retrieve all of the training sequence successfully. This happens when there are frequent occurrences of repeating patterns, as in sequences $ABCABDBE\cdots$ and $QQXQQYQQZ\cdots$.

We have assumed so far that each HANN in the system learns heteroassociations between single spatial patterns only, and does not learn to associate multiple patterns in groups. For example, it learns to associate pattern $A$ with pattern $B, E$ with $H$, and so on; however, it does not learn to associate $AAB$ with $A$, and so on. By relaxing this single pattern assumption and by allowing the HANN’s to learn heteroassociations between groups of multiple spatial patterns, the system will have an increased ability to handle sequences with many repeating patterns. For example, $ABCABDABE\cdots$ and $QQXQQYQQZ\cdots$ can be learned by associating $ABC$ with $A$, $QQX$ with $Q$, and so on. However, the system will then need groups of multiple spatial patterns as cues during retrieving, and will have reduced robustness against temporal gaps in cue sequences. For example, if sequence Fig. 1(b) is also stored in such a system, cue sequences $5\square\square$ will not lead to a successful retrieval, where $\square$ denotes a temporal gap, since the system has not learned group $5\square\square$ during training. Similarly, since the system is not able to associate single patterns with single patterns, a single pattern will also be unable to retrieve sequence Fig. 1(b). In comparison,
a system with three HANN’s that learns associations between single patterns only is able to use cues such as 5□□ and 2 to retrieve sequence Fig. 1(b), as shown in Fig. 4(a), (f), and Tables IV and IX.

We stress that the present system learns and generates spatial–temporal sequences; however, it does not recognize them. For example, after the system learns sequences (a)–(c) in Fig. 1, it is able to generate them when presented with cues during retrieving. But the system does not know that sequence (a) means “food” or that sequences (b) and (c) mean “danger.” However, the system may be useful, for example, to an animal if the earlier part of sequence (a) is related to “food” and the latter part of sequence (a) is related to a “feeding” routine, whereas the earlier parts of sequences (b) and (c) represent dangerous scenes and the latter parts are escape sequences.

In Section V, we have presented a special case of our general system for spatio-temporal sequence storage and retrieval. The general system can be implemented using any other static heteroassociative neural networks, such as the backpropagation network [26] and the bidirectional associative memory [20]. Another special case of the system was discussed in [39].

In general, the neurons in a HANN may output continuous (analog) values instead of binary or discrete (digital or gray scale) values. For example, the original backpropagation network and the original counterpropagation network, a variation of which has been discussed in the previous section, all output continuous values. Although we have used sequences consisting of binary patterns in our examples, the present system is not restricted to patterns with discrete values.

We note that it is possible to construct a HANN from an AANN by connecting two such AANN’s in a way so that a pattern $P_1$ in one of the two AANN’s invokes another pattern $P_2$ in the other AANN [27], thereby achieving heteroassociation. Hence, it is possible to use any arbitrary static associative neural network, no matter whether autoassociative or heteroassociative, for spatio-temporal sequence generation in the present framework.

VII. Summary

We have presented the design of a general system that can learn and generate spatio-temporal sequences, using any static associative neural networks. After learning, the system is able to retrieve the whole sequence after being presented with a small piece, which may or may not have spatial imperfections and/or temporal gaps, of a stored sequence. This is equivalent to, after learning and when a sequence of events is presented to the system, the system predicting the sequence of events in the future. As a demonstration of the applicability of the present general system, we have presented an implementation using our dynamically generated variant of the counterpropagation network. This system shows a number of desirable properties, such as short learning time, fast and accurate retrievals, and the ability to store a large number of sequences consisting of nonorthogonal spatial patterns. These are important for practical applications such as real-time speech production and robotic control.

ACKNOWLEDGMENT

The author thanks the reviewers for very carefully reading the paper, and for offering numerous helpful suggestions and comments which have led to significant improvement of this work. The author also thanks A. Goodman for technical editing.

REFERENCES


Lipo Wang received the B.S. degree in laser optics from the National University of Defense Technology, Changsha, China, in 1983, and the Ph.D. degree in physics from Louisiana State University, Baton Rouge, in 1988 (with a CUSPEA assistantship). Since then, he has been interested in the theory and applications of neural networks. In 1989, he worked at Stanford University, Stanford, CA, as a Post-Doctoral Fellow. In 1990, he was a Lecturer in electrical engineering at the University College of University of New South Wales, Australia. From 1991 to 1994, he was on the research staff at the National Institute of Health, Bethesda, MD. Since 1995, he has been with the School of Computing and Mathematics, Deakin University, Melbourne, Australia, where he is a tenured Computing Lecturer. His research interests include temporal behavior, pattern recognition, optimization, control, data mining, fuzzy networks, and biological networks. He is the author or co-author of 40 journal publications and 30 conference presentations, holds a U.S. patent on neural networks, and is the co-author/editor of Artificial Neural Networks: Oscillations, Chaos, and Sequence Processing (Los Alamitos, CA: IEEE Computer Society Press, 1993). He is an associate editor for Knowledge and Information Systems: An International Journal, and has served on the program committees of a number of conferences.