Anticipation-Based Temporal Pattern Generation

DeLiang Wang, *Member, IEEE,* and Budi Yuwono

***Abstract*—A neural network model of complex temporal pattern generation is proposed and investigated analytically and by computer simulation. Temporal pattern generation is based on recognition of the contexts of individual components. Based on its acquired experience, the model actively yields system anticipation, which then compares with the actual input flow. A mismatch triggers self-organization of context learning, which ultimately leads to resolving various ambiguities in producing complex temporal patterns. The architecture of the model incorporates a short term memory for building associations between remote components and recurrent connections for self-organization and component generation in a temporal pattern. Synaptic modification is based on a one-shot normalized Hebbian rule, which is shown to exhibit temporal masking. The major conclusion, namely the network model can learn to generate any complex temporal pattern, is established analytically. An estimate on the efficiency of the training algorithm is provided. Multiple temporal patterns can be incrementally acquired by the system, exhibiting a form of retroactive interference. Neural and cognitive plausibility of the model is discussed at the end of the paper.**

I. Introduction

THE ability to learn and generate temporal patterns is one of the most important characteristics of an intelligent system. In fact, it is a necessary survival ability for many animals, such as in recognizing dangerous scenes, escaping from enemies, etc. Such an ability enables the systems to perform tasks, ranging from a simple behavior of limb movement to abstract temporal reasoning. Time may be embedded in a temporal pattern in two basic ways: 1) *Temporal order.* If the components of a temporal pattern are drawn from a specific alphabet, temporal order refers to the ordering among a set of symbols from the alphabet (a sequence). For example, the sequence A-B-C is considered different from C-B-A because of different ordering. Temporal order may also refer to a syntactic structure, such as subject-verb-object, where each component may be any of a number of possible symbols. 2) *Time duration.* Time duration is inversely proportional to the rate of presentation. Duration plays a critical role for temporal processing, both in recognizing and producing temporal patterns.

To focus the following discussions, we assume discrete temporal patterns, or temporal sequences. Of course, a temporal pattern may be continuous, and in this case, it can usually be sampled into a sequence of discrete patterns before processing. Following the terminology introduced by Wang and Arbib [40], a sequence S of length N over a symbol alphabet is defined as $p_1 - p_2 - \cdots - p_N$, where each p_i $(1 \leq i \leq N)$ is one symbol of the symbol alphabet and a component (static pattern) of S. Any part of S, $p_j - p_{j+1} - \cdots - p_k$ where $1 \leq j \leq k \leq N$, is called a subsequence of S. Generally, in order to unambiguously produce a component in a sequence, a prior subsequence is required to be detected. For instance, in the sequence R-E-F-E-R-E-E, the prior subsequence E-R-E of the last E is required to determine the E, since R-E is a recurring subsequence in the sequence. Thus we define the *context* of a component p_i as the shortest subsequence which unambiguously determines p_i in sequence S, and its length is called the *degree* of p_i. The *degree of a sequence* is the maximum degree of its components. Thus, a *simple* sequence that contains no recurring subsequences corresponds to a degree 1 sequence. A sequence is *complex* if its degree is greater than one.

Neural networks to learn and generate temporal sequences have been investigated by a number of investigators (See Wang [39] for a more extensive review). Perhaps the earliest model of sequence generation is the *outstar avalanche* by Grossberg [13], which is composed of a number of sequential outstars corresponding to static patterns. Each outstar is connected with the next one in the sequence, leading to a sequential recall of static patterns once the first pattern is activated. Based on the idea of interconnecting two networks, Kosko [23] proposed a bidirectional associative memory model that can generate a sequence of patterns which alternate between the two memory networks. Along a similar line, Healy *et al.* [16] coupled two ART 1 modules [5] and associated the pattern learned from one ART 1 to another one.

Associative memory models have been extended to store and produce temporal sequences (see among others [36], [3], [15], [18]). Here, a sequence is treated as a set of pairs between consecutive components, and these pairs are stored into an associative memory. Hence, after the first component of the sequence is presented, the next component will be activated from the memory shortly, which further activates the third one, etc. This process continues until the entire sequence is produced. Generation of temporal sequences have also been investigated using the backpropagation network [21], [10], [11]. There are two basic styles of architecture: In the Jordan network [21] the output layer associated with a component is fed back and blended with some history to generate the next component, whereas in the Elman network [11] the hidden layer is fed back to influence the generation of the next component.

One of the main problems with the above methods lies in producing complex sequences, where one pattern may be followed by different ones. In this situation, it is insufficient

Manuscript received December 22, 1993; revised June 3, 1994. This work was supported in part by NSF grant IRI-9211419 and ONR grant N00014-93-1-0335.

The authors are with the Laboratory for AI Research, Department of Computer and Information Science and Center for Cognitive Science, The Ohio State University, Columbus, OH 43210-1277 USA.

IEEE Log Number 9406633.

to have just one prior component to cue a current one. High-order networks have been proposed to fix the problem [9], [15]. In a high-order network, one component in a sequence is associated by a prior subsequence of certain length depending on the order of the network. The major concern with high order networks is the system overhead caused by the immense number of connections. The number of connections in a high-order network grows exponentially with the order of the network. Another difficulty is that unless the degree of the sequence is known in advance, a fixed network cannot guarantee unambiguous production of an arbitrary sequence.

Recently, Wang and Arbib [40], [41] proposed a model for temporal sequence learning. Complex temporal sequences are acquired by a network through a form of supervised learning, called attentional learning. The basic assumption for sequence reproduction is that the generation of the next component is based on the recognition of its context. Thus, after successful training, an adequate beginning part of a sequence forms the context for activating the next component. This newly activated component joins the previous context, which is fading as time goes, to form another context which will be able to trigger yet next component. This process continues until the entire sequence is reproduced. Based on this idea, the later model [41] addressed the issues of time warping and chunking of subsequences. In particular, sequences can be recognized in a hierarchical fashion and without being affected by the presentation speed. Also in sequence generation, relative timing among the sequence components can be maintained by the model.

This paper presents a neural network model of learning and generating complex temporal patterns by self-organization. The basic idea for self-organization is an *anticipation mechanism* where the system actively anticipates the next component in a sequence and compares its anticipation with the next input component. The self adjustment of contexts is performed once the anticipation does not match the next input. We introduce a layer of neural modulators which perform the system anticipation. Based on this mechanism, we show that the network can learn any complex sequences. The present model differs from the earlier models proposed by Wang and Arbib [40], [41] in a number of major aspects. First, the selection of context detectors is done by competitive learning in this model whereas the previous models need the system to assign and remember appropriate context detectors. Secondly, active anticipation is missing from the previous models. Combined together, the present model fully organizes itself during sequence training and learns to generate arbitrarily complex sequences. Also a much more extensive analytical investigation has been undertaken for the present model.

The remaining part of the paper is organized as follows. In Section II, we describe the architecture of the neural network model. Section III presents the definition of model dynamics and the learning algorithm used by the network. In Section IV the model is formally analyzed, and we summarize the main result as a theorem which guarantees that the model can learn to generate any complex sequence. In Section V, the efficiency of the learning algorithm and acquisition of multiple sequences are analyzed. The model is shown to exhibit incremental learning, whereby later sequences can be acquired without damaging memory traces of previously learned sequences. We conclude the paper with some general discussions in Section VI about possible extensions of the model and its link to neurobiological and psychological data.

II. Network Architecture

As a basic idea, this model generates a sequence S of length N by successively predicting component p_i based on the context of p_i. More specifically, the model is designed to detect the context of p_i first, and then associate this detection with p_i, thus producing p_i once its context occurs from the input. Since a context is, by definition, also a sequence, recognizing the contexts of sequence components can be done by means of a set of sequence detectors, each of which is uniquely associated with a component. In the following discussions, we assume that each symbol, or static pattern, is uniquely represented by an input unit. This assumption is introduced to focus the attention to processing temporal aspects in sequence learning. In general, some pattern recognition networks, such as ART [5], [6] or self-organizing maps [22], can serve as the front-end to this network for recognizing spatial (static) patterns from the input flow.

In order to learn a complex sequence, each input component must be associated with other successive components beyond its immediate successor in the sequence. This can be achieved with a short-term memory (STM) model (see [40], [41], [2], [31]). In a decay-based STM model, when component p_i is presented, the corresponding input unit U_i is activated. Assuming discrete time steps, the activity level of U_i then decays after each time step, resulting in a pattern of activity where the more recent a component is presented the higher the activity level of its corresponding input unit. In other words, a sequence of successive input patterns are made simultaneously available as a spatial pattern due to STM. The resulting spatial pattern is a vector V whose elements represent sequence components, so that the magnitude of each vector component corresponds to the temporal order of the corresponding sequence component. All input units together form the input layer.

In order to implement such an STM model, each input unit is extended to a local network, called a shift-register assembly. A *shift-register assembly* is a group of units, called shift-register units, which are serially linked one after another, forming a chain like the outstar avalanche [13]. Fig. 1 shows such an assembly. Each assembly is triggered externally through a *head* unit at one end of the register-chain. The head unit receives its input from a unique input *terminal* which is directly activated by external stimulation or by a modulator unit (to be discussed shortly) during generation. Each terminal represents a unique input symbol. When a symbol is presented to the network, the corresponding terminal is activated. When a terminal is triggered, the head unit of the respective assembly is activated. The activity of each unit in the assembly is then shifted to the next one along the chain in the next time step. Every time an activity is shifted, its magnitude is reduced by a constant amount until the magnitude is reduced to 0.

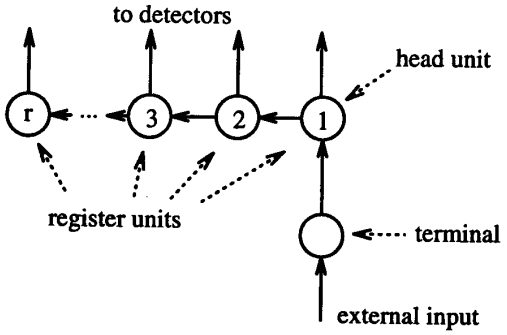


Fig. 1. A shift-register assembly. The right-most register unit, called the head unit, receives activation from a corresponding input terminal. The activity of a register unit decays and is shifted to its left neighbor every time step.

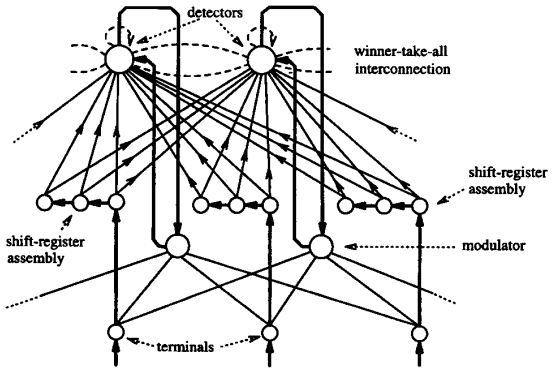


Fig. 2. The architecture of the overall network model. Thin lines denote modifiable connections, while thick lines and dashed lines denote fixed connections.

With such a STM model in place, a single detector layer with m units is introduced for detecting different contexts. Fig. 2 shows the entire neural network architecture of this model. Each unit in this layer receives inputs from all the shift-register units (SR units) and this layer can be trained to associate its units with the input subsequences corresponding to different contexts. A detector in the network is said to be *committed* if it is tuned to a particular input subsequence, i.e., the detector unit will be activated when the subsequence is presented. We use a winner-take-all mechanism for the detector layer as described by Grossberg [14]. In this mechanism, each unit inhibits all the others in the network, and it also excites itself (see Fig. 2, dashed lines). The overall effect of this competitive network is that the unit with the highest initial activity is the only one that has certain activity while all the others are inactive (with 0 activity). Due to winner-take-all dynamics, there is at most one detector unit committed to a particular subsequence.

The system anticipates the next component and compares it with the external input through the modulator layer. As shown in Fig. 2, there is one modulator associated with each detector in the network. Each modulator unit receives upward connections from every individual terminal. In addition, it receives a downward connection from its respective detector. An active detector enables its modulator in the next time step (assuming some delay). Once enabled, the modulator performs one-shot (single step) learning which updates its connection weights from the terminals. Since only one terminal corresponding to an input component can be active at any time step, one-shot learning leads to one-to-one connection from an active terminal to a triggered modulator. Basically, this one-shot learning establishes the association between a context detector and the next input component. If they do not match next time when the detector is activated, the anticipated activation of the corresponding modulator will be absent, and this mismatch will be detected by the modulator, which in turn sends a signal to its respective detector to expand the context that the detector is supposed to recognize. We provide the detailed neural dynamics and the learning algorithm in the next section.

III. Learning Algorithm

A. Model Definition

Let us assume that the network model has m detectors, m modulators, n terminals and n shift-register assemblies. Each assembly has r SR units. The activity of detector i at time t, $E_i(t)$, is defined as:

$$E_i(t) = g\left(\sum_{j,k} W_{i,jk} V_{jk}(t), \theta_i\right) \tag{1}$$

$$g(x,y) = p\begin{cases} x & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $W_{i,jk}$ is the connection weight from the kth SR unit of assembly j to detector i. $V_{jk}(t)$ is the activity of this SR unit at time t. θ_i is an adjustable threshold for the detector, which is initialized to 0. θ_i may be increased when detector i wins the winner-take-all competition in the detector layer, to be discussed later. The activity $V_{jk}(t)$ is defined as follows,

$$V_{jk}(t) = \begin{cases} I_j(t) & \text{if } k = 1 \text{ (head unit)} \\ \max\left[0, V_{j,k-1}(t-1) - \delta\right] & \\ & \text{otherwise} \end{cases} \tag{3}$$

where $I_j(t)$ is the binary activity of terminal j. $I_j(t) = 1$ if the corresponding symbol of terminal j is being presented to the network at time t, and $I_j(t) = 0$ otherwise. Due to the nature of sequential input, at most one terminal has its $I(t)$ equal to 1. δ is a decay parameter. Eq. (3) provides an implementation of the STM model described earlier, i.e., an input activity is held for a short time and decays gradually in a shift-register assembly. If assembly j is stimulated by an input at time t, namely $I_j(t) = 1$, according to (3) the end unit of the assembly gets activated at time $t + r - 1$, and its activity $V_{jr}(t + r - 1) = 1 - \delta(r - 1)$. Apparently, the input cannot be held longer than r steps, the limit of STM capacity. Given r, in order for the input to be held for r steps, the parameter must be chosen so that $1 - \delta(r - 1) > 0$. That is, $\delta < 1/(r - 1)$.

It was mentioned in the previous section that all the detector units in the detector layer form a winner-take-all network. The detailed dynamics of winner-take-all can be found in [14]. In such a competitive network, the activity of each detector evolves until the network reaches equilibrium, where

the detector with the highest initial activity becomes the only active unit. The network takes a short time to reach equilibrium. This time period should be much shorter than the duration of one sequence component[1] In the following analysis, we assume that each discrete time step is longer than the time needed for the winner-take-all mechanism to reach an equilibrium.

The learning rule for each detector i is a Hebbian rule [17] plus normalization to keep the overall weight a constant [25], [40], denoted as a *normalized Hebbian rule*,

$$\hat{W}_{i,jk}(t+1) = W_{i,jk}(t) + \alpha O_i(t) g(V_{jk}(t), A_i) \quad (4a)$$

$$W_{i,jk}(t+1) = \frac{\hat{W}_{i,jk}(t+1)}{\alpha C + \sum_{j,k} \hat{W}_{i,jk}(t+1)} \quad (4b)$$

where α is a gain parameter or learning rate. A large α makes training fast. It is easy to see that when α is very large, approximate one-shot learning is exhibited as a result. As mentioned earlier, winner-take-all competition in the detector layer will activate a single unit from the layer. To indicate the outcome while omitting the details of competitive dynamics, we introduce $O_i(t)$ which equals 1 if detector i is the winner of the competition, or 0 otherwise. Function g, as defined in (2), serves as a gate to let in the influences of only those SR units whose activities are greater than or equal to A_i. A_i is the sensitivity parameter of unit i. The lower the sensitivity parameter the more SR units can be sensed by a winning detector. Thus more connections of the detector can be modified according to (4a). Furthermore, the sensitivity parameter A_i is adaptive by itself, following the rule below:

$$A_i = \begin{cases} 1 & \text{if } d_i = 0 \\ \max[0, 1 - \delta(d_i - 1)] & \text{if } d_i > 0 \end{cases} \quad (5)$$

where d_i, called the *degree parameter* of detector i, is a non-negative integer, initialized to 0. δ is the decay parameter as introduced in (3). According to (5), A_i is equal to 1 when d_i = 0 or 1, and decreases until 0 as d_i increases. Since value 1 is the activity level of the corresponding head unit when some assembly is stimulated, detector i will only sense a head SR unit when d_i = 0 or 1. When d_i increases, more SR units can be sensed, and except when d_i = 0, d_i is equal to the number of units that detector i can sense when it becomes a winner. The constant C in (4b) is positive, and its role will be analyzed in Section IV. The connection weight $W_{i,jk}$ is initialized to $1/[r(1 + C) + \varepsilon]$, where ε is a small random number introduced to break symmetries between the inputs of the detectors, which may cause problems for competitive dynamics.

Let us denote unit z as the winner of the competition in the detector layer. After the connection weights are updated, the activity of unit z will change as a result with the same input in the future. More specifically, E_z is monotonically non-decreasing as learning takes place. This observation will be proven in the next section. This resulting (increased) activity by (1) is then used to update the threshold of unit z. This can be generally described as:

$$\theta_i(t+1) = \theta_i(t) + O_i(E_i^*(t+1) - \theta_i(t)) \quad (6)$$

where $E_i^*(t+1)$ is the activity of i based on the new weights, i.e. $E_i^*(t+1) = \Sigma_{jk} W_{i,jk}(t+1) V_{jk}(t)$. Thus, θ_i is adjusted to E_i^* if unit i is the winner. Otherwise, θ_i remains the same. Due to this adjustment, unit z increases its threshold so that unit z will be triggered only by the same input subsequence whose components have been sensed during weight updates by (4a). This assertion will be proven in the next section. The above threshold can be relaxed (lowered) a little in order to handle sequences with certain distortions. This way, subsequences very close to the training one can also activate the detector.

A modulator receives both top-down connection from its corresponding detector and bottom-up connections from input terminals (Fig. 2). We assume that the top down connection *modulates* the bottom-up connections by a multiplicative operation. Thus, the activity of modulator i is defined as,

$$M_i(t) = O_i(t-1) \sum_{j=1}^{n} R_{ij} I_j(t) \quad (7)$$

where R_{ij} is a binary-valued weight of the connection from terminal j to modulator i. All R_{ij}'s are initialized to 0. It is assumed that the top-down signal from the detector takes one time step to reach its modulator. Since at any time, at most one terminal is active ($I(t)$ = 1), $M_i(t)$ is also a binary value. If $O_i(t-1)$ = 1 and $M_i(t)$ = 0 then the modulator sends a feedback signal to its detector. Upon receiving this feedback signal, the detector increases its degree parameter, thus lowering its sensitivity parameter A_i (see (5)). Quantitatively, d_i of detector i is adjusted at time t as follows:

$$d_i = \begin{cases} d_i + O_i(t-1) & \text{if } M_i(t) = 0 \\ d_i & \text{otherwise} \end{cases} \quad (8)$$

We refer to this situation where $O_i(t-1)$ = 1 and $M_i(t)$ = 0 as a *mismatch*. A mismatch occurs when an anticipated next component in the sequence does not appear, to be explained shortly. Thus the degree of a context detector increases when a mismatch is caused by the detector.

Finally, one-shot learning is performed on the bottom-up connection weights of the modulator of the winning detector z,

$$R_{zj} = I_j(t) \quad (9)$$

This learning rule sets the connection weights of modulator z to the current activities of the input terminals. Since there is only one active terminal at time t, i.e., the one representing the current input symbol, only one bottom-up weight of the modulator is equal to one, and all the others are zero. The result of this training is a one-to-one association between a modulator and a terminal.

Let us return to explain under what condition a mismatch occurs, which leads to an increase of the degree of detector z. According to (7), a mismatch occurs when $O_i(t-1)$ = 1 and $\Sigma_{j=1}^{n} R_{ij} I_j(t)$ = 0. Since at any time, only one bottom-up weight of modulator i equals 1 and only one input terminal (I_j) is activated, mismatch occurs when the

[1] From the neurophysiological perspective, the kind of neural dynamics that occurs in the winner-take-all network should take no more than dozens of milliseconds, while typical perception of a sequence component should take hundreds of milliseconds.

non zero weight and the terminal with non zero input do not coincide with each other. But one-shot learning of (9) establishes a non zero link only between a modulator and the next input terminal. Therefore, if the link between detector i and an input terminal established last time when detector i was activated does not coincide with the activated input terminal this time (at time t), a mismatch occurs. The bottom-up links of a modulator established between the modulator (or the corresponding detector) and the next input component are used for the modulators to anticipate the next component in sequence generation. Thus a mismatch corresponds to the situation where the anticipated next input symbol does not match with the real input symbol during sequence training. Since R_{ij}'s are all initialized to 0, following (7) a mismatch is bound to occur the first time a pair of consecutive components is presented, which then increases the degree of the detector for the first component from 0 to 1. If the sequence to be learned is a simple sequence, like *A-B-C-D-E*, it suffices to increase the degrees of all detectors involved to 1. For complex sequences, though, the degree parameters of the relevant detectors will further increase until no mismatch occurs.

The training steps are repeated at each following time step. After all sequence components have been presented, the entire cycle of training, referred to as a *training sweep*, is repeated. The training phase is completed when there is no mismatch occurring during the last training sweep. In this case, as shown in the next subsection, the network correctly anticipates the next component for every time step. The completion of the learning phase can be detected in various ways. For example, a global unit can be introduced to sum up all feedback from modulators to their respective context detectors during a training sweep. Thus, the inactive global unit by the end of a sweep signals the end of the learning phase.

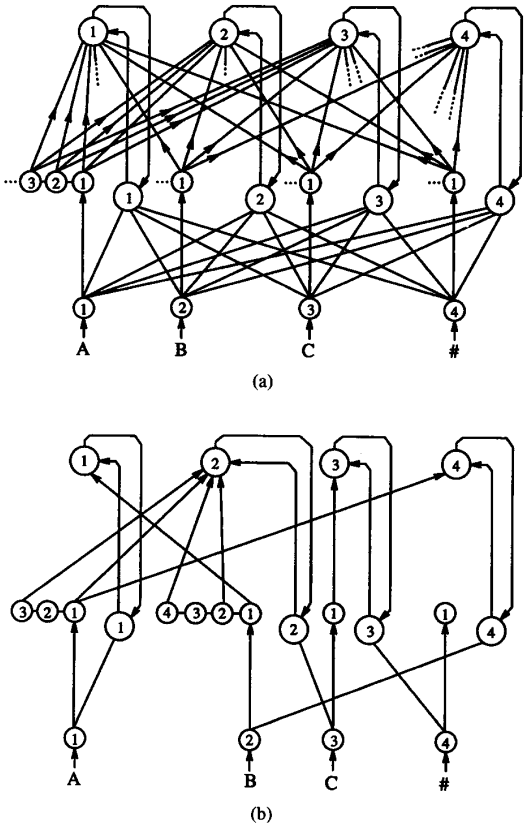


Fig. 3. (a) An example network configuration. Each component of the input sequence is presented to the network by triggering the input terminal representing the corresponding symbol. (b) The resulting network configuration after successful training with the sequence S_e.

B. Sequence Generation

During sequence generation (reproduction), the connections from the input terminals to the modulators in the model are reversed (see Fig. 2). Since only one bottom-up link from the terminals to a modulator is non-zero after one-shot learning of (9), once reversed, a modulator triggers only one terminal. The generation process of a learned sequence starts from the presentation of the beginning component of the sequence. The beginning component will trigger an appropriate context detector, which in turn activates its respective modulator, thus leading to the activation of the second component in its corresponding terminal. The activated terminal joins the beginning component to activate another context detector, which again triggers its respective modulator and thus the third input component. This process continues until the entire sequence is generated.

The reversal of the connections from the terminal layer to the modulator layer can be neurally implemented by introducing bidirectional connections between the two layers and assuming that the backward connections from the modulator layer to the terminal layer are trained in the same way as the forward connections. Thus the backward connections established during the learning phase are used directly during the generation phase. A similar idea has been proposed for training bottom-up and top-down connections simultaneously in the ART 1 network [5].

In the present model, the duration during which a sequence component is presented is not learned by the network. Therefore, the components generated are of equal duration. To encode duration, the model can be modified by introducing a weighted connection from each detector to its modulator. The weight of this connection can serve as a propagation delay, and can be adjusted to the duration of a corresponding component. This connection weight prevents the detector from triggering the modulator until an appropriate propagation delay has elapsed. A similar idea has been proposed by Wang and Arbib [41] to produce durations of sequence components.

C. Example

Let us see how the algorithm operates by examining an example sequence, which is a complex sequence S_e = *B-A-B-A-C*. Let terminals 1, 2, 3, and 4 represent symbols *A*, *B*, *C* and # respectively, where '#' is the end marker of a sequence. Fig. 3(a) shows the network configuration for this example. All units in the detector layer are uncommitted at the beginning.

During the first training sweep, the first component B is presented to the network at time 0 by activating terminal 2. Let us assume that detector 1 wins winner-take-all competition. Weight update is then performed by detector 1 according to (4), followed by an adjustment to its threshold θ_1 according to (6). Next, detector 1 sends its output $O_1(0)$, which equals 1, to modulator 1, and this signal arrives at the modulator at time 1. At time 1, symbol A is presented to the network by activating terminal 1. At this time, modulator 1 computes its activity level according to (7), resulting in 0 since all R_{ij}'s are initialized to 0. Because $M_1(1) = 0$, a feedback signal is sent to detector 1 to increase d_1 (see (8)). Meanwhile, modulator 1 performs one-shot learning according to (9), and thus set $R_{1,1}$ to 1 and $R_{1,j}$ ($j \neq 1$) to 0. At this point, detector 1 is tuned to input subsequence B and anticipates the next component, symbol A.

At time 1, the next winner from the detector layer is selected. Notice that the previous winner, detector 1, can not be the winner at this time because θ_1 is now set to $E_1(0)$. Without loss of generality, let the current winner be detector 2. Detector 2 performs weight update by (4) and then sets its threshold θ_2 to $E_2(1)$. At $t = 2$, the second B is presented by activating terminal 2. Modulator 2 performs one shot learning of (9), resulting in $R_{2,2} = 1$ and $R_{2,j} = 0$ for $j \neq 2$. Also d_2 is increased to 1. At this point, detector 2 is tuned to subsequence A and its modulator anticipates the third component of S_e. With symbol B active, detector 1 becomes a winner again, since the learning rule of (4) makes the detector even more competitive than when the first B was presented. Detector 1 performs weight updates again, and then adjusts its threshold.

When $t = 3$, terminal 1 representing symbol A is again stimulated, and modulator 1 results in $M_1(3) = 1$. Since $M_1(3)$ is not zero, no mismatch occurs. At time 3, detector 2 is the winner of the detector layer and activates its modulator. At time $t = 4$ terminal 3 representing symbol C is now activated, not terminal 2 (symbol) which modulator 2 anticipates. Thus a mismatch occurs. Modulator 2 sends a feedback signal to detector 2 to increase its degree to 2. Modulator 2 also performs one-shot learning, adjusting $R_{2,3}$ to 1 and $R_{2,j} = 0$ for $j \neq 3$. Now, detector 2 detects A but anticipates C. Meanwhile, an uncommitted detector unit, let it be detector 3, becomes a winner, which is tuned to subsequence C. Modulator 3 then establishes a link with #. Thus d_3 is set to 1. By now, the first training sweep is completed.

At the beginning of the second sweep, let time be reset to zero for the convenience of discussion. At time $t = 0$, symbol B is presented by activating terminal 2. At time $t = 1$, terminal 1 is activated. No mismatch occurs. At the same time, detector 2 becomes a winner and updates its weights with $d_2 = 2$. The resulting connection weight distribution after (4) is tuned to B-A. Also θ_2 is increased according to (6). At time 2, terminal 2 is stimulated. In this step, modulator 2 finds a mismatch because terminal 2 instead of terminal 3 which represents C is currently stimulated. Thus, d_2 increases to 3. Then, modulator 2 updates its bottom connections so that $R_{2,2} = 1$ and $R_{2,j} = 0$ for $j \neq 2$. Furthermore, detector 1 becomes a winner at time 2. At $t = 3$ when symbol A is presented, detector 2 which is tuned to detect B-A becomes a winner. Now, detector 2 learns to detect A-B-A. However, in the next time step ($t = 4$), terminal 3 is stimulated, instead of terminal-2 which modulator 2 anticipates. Thus a mismatch occurs and d_2 is increased to 4. Also modulator 2 anticipates terminal 3. At time 4, detector 3 is a winner. For time 0 of the third sweep, training proceeds as before. At time $t = 1$ when terminal 1 is stimulated, detector 2 which is tuned to A-B-A, is prevented from winning by its threshold θ_2 which was increased during its previous training. As a result, an uncommitted detector unit, let it be unit 4, becomes a winner, and is tuned to A after training. At time $t = 2$, detector 1 becomes a winner. However, modulator 4 establishes a link to terminal 2. At time $t = 3$ when terminal 1 is stimulated, detector 2 which is tuned to A-B-A becomes a winner, while detector 4 which is tuned to A cannot win due to its shorter context. At time $t = 4$, training proceeds as in the previous sweep.

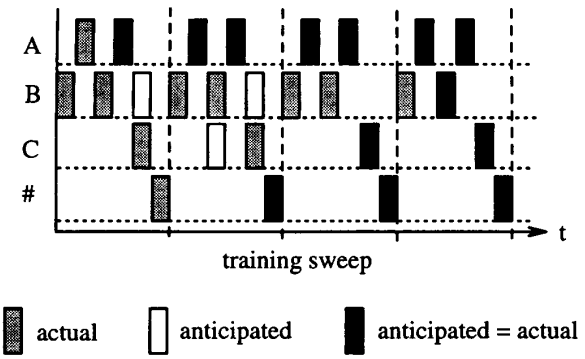


Fig. 4. The activity traces of the input terminals of a network during training of sequence S_e. A white box represents an actual terminal activity, a gray box represents an anticipated activity, and a black box represents a match between an anticipated and an actual input activity. The training phase was completed after 5 sweeps. The simulated network consists of 24 detectors, 24 terminals, and 6 SR units for each shift-register assembly. The parameter values used are: $a = 0.2$, $d = 0.1$, and $C = 3.0$.

The fourth training sweep proceeds with no mismatch found, and therefore the training phase is completed after this sweep. The resulting network configuration is shown in Fig. 3(b), where unit 1 in the detector layer detects the context B and anticipates A, unit 2 detects the context B-A-B-A and anticipates C, and so on. Fig. 4 shows the activity traces of the four terminals of Fig. 3 during the learning phase. Once learned, the sequence can be generated by presenting the beginning component B to the network.

IV. Analysis of the Algorithm

We now analytically establish a number of properties of the above model for sequence generation. In the end, we conclude with a theorem which states that the model can learn to generate any complex sequences, provided that the number of SR units in each assembly is greater than or equal to its degree of the sequence. Let us point out that the degree of a sequence should not exceed r, the number of SR units in each assembly. Otherwise, the sequence cannot be acquired. This is because the maximum degree of the contexts that can be held in STM is equal to r, not large enough to resolve ambiguities in a sequence whose degree is greater than r. Thus, the selection

of r limits the degree of the sequence that can be learned. To simplify the analysis, we assume one shot learning, which can be made by choosing an appropriate α in (4a). We also assume that there is sufficient supply of uncommitted units in the detector layer.

Before we present the following proposition, let us define that a right subsequence of sequence S is a nonempty right part of S. For example, sequences D, C-D, B-C-D are all the right subsequences of sequence A-B-C-D. Similarly, we define a sequence that has S as a right subsequence as a left supersequence of S. For example, sequences C-D, B-C-D, and A-B-C-D are some of the left supersequences of D. The next right subsequence of S is a right subsequence of S whose length is 1 minus the length of S. Similarly, the next left supersequence of S is a left supersequence of S whose length is 1 plus the length of S. For example, B-C-D is the next right subsequence of A-B-C-D, and A-B-C-D is the next left supersequence of B-C-D.

Proposition 1: The learning rule of (4) with a proper choice of parameter C guarantees that the detector of sequence S is preferred to the detectors of all the right subsequences of S.

In other words, when sequence S occurs, the detector that recognizes S masks those detectors that recognize the right subsequences of S. We call this property of the learning rule *temporal masking*, following the term masking fields introduced by Cohen and Grossberg [8], which states that a larger spatial pattern is preferred to a smaller one in activating their corresponding detectors.

Proof: Let S be $p_1 - p_2 - \cdots - p_d$. When S has been presented completely, the SR activity corresponding to p_i is $1-(d_i)\delta$ $(i = 1, \cdots, d)$, given that $1-(d-i)\delta \geq 0$ (otherwise $1-(d-i)\delta$ becomes 0, see (3)), and the overall activity $E(d)$ as the function of d

$$E(d) = \sum_{i=1}^{d}[1-(d-i)\delta] = d[1-(d-1)\delta/2] \tag{10}$$

Let unit z be the detector trained for recognizing S. Thus the degree of z is d. Due to one shot learning, the weight W_i from the SR unit representing p_i to z equals $[1-(d-i)\delta]/(C+E(d))$. Therefore, at this time,

$$\begin{aligned} E_z(d) &= \sum_{i=1}^{d} W_i[1-(d-i)\delta] \\ &= \sum_{i=1}^{d}[1-(d-i)\delta]^2/(C+E(d)) \end{aligned} \tag{11}$$

Now, proposition 1 becomes equivalent to that a proper C can be selected so that

$$E_z(d) > E_z(d-1) \tag{12}$$

where $E_z(d-1)$ is equal to the activity of the detector for the next right subsequence of S, $p_2 - \cdots - p_d$. According to (11) and (10), (12) is equivalent to

$$\begin{aligned} C+d-1-(d-1)(d-2)\delta/2 > \frac{1}{1-(d-1)\delta} \\ \cdot\{(d-1)-\delta(d-1)(d-2) \\ +\delta^2[1^2+2^2+\cdots+(d-2)^2]\} \end{aligned}$$

TABLE I
SELECTION OF C WITH RESPECT TO δ AND r

C_{min}		r			
		2	3	4	5
δ	0.1	0.1	0.11	0.36	0.8
	0.2	0.25	0.93	2.6	9.0

According to the equation

$$\sum_{i=1}^{q} i^2 = \frac{2q^3+3q^2+q}{6} \tag{13}$$

We obtain

$$\begin{aligned} C+d-1-(d-1)(d-2)\delta/2 > \frac{1}{1-(d-1)\delta} \\ \cdot\left\{(d-1)[1-(d-2)\delta]\right. \\ \left.+\frac{\delta^2}{6}(d-1)(d-2)(2d-3)\right\} \end{aligned}$$

After simplifying and rearranging the above terms, we finally have

$$C > \frac{\delta d(d-1)}{6}\left[1+\frac{\delta+2}{1-\delta(d-1)}\right] \tag{14}$$

Since the degree of the sequence that is to be learned cannot be greater than r, the number of the SR units in a shift-register assembly, as long as C is chosen to be

$$C > \frac{\delta r(r-1)}{6}\left[1+\frac{\delta+2}{1-\delta(r-1)}\right] \tag{15}$$

inequality (12) will be true. Q.E.D.

Inequality (15) tells us how to choose C based on the value of δ in order to ensure that the detector of a sequence is preferred over the detectors of its right subsequences. It can be seen that the smaller is δ, the smaller is the right-hand-side of (15), and thus the smaller a value of C can be chosen to satisfy the inequality. As a degenerate case, if $\delta = 0$, (15) becomes $C > 0$, and this corresponds to the condition of forming masking fields in static pattern recognition [8]. Therefore, (15) includes masking fields as a special case. In temporal processing, δ reflects forgetting in STM, and thus cannot be 0. On the other hand, δ should be smaller than $1/(r-1)$ in order to fully utilize available SR units for STM (see the discussions following (3)), thus the degree of the learnable sequences (see (3)). Table I gives an idea about the relation between $C_{\min}$, the minimum C satisfying (15), r, and δ.

In general we define function $h(d)$ as

$$h(d) = \sum_{i=1}^{d} [1 - (d - i)\delta]^2/(C + E) \tag{16}$$

the same as $E_z(d)$ in (11). Also we refer to the situation when a detector is chosen for the first time as *initial training*.

Proposition 2: Initial training of a detector results in only one nonzero weight for the detector, which equals $1/(1 + C)$. Also, the activity of the unit equals} $h(1)$ and the threshold increases to $h(1)$.

Proof: Let the unit in question be z. For initial training $\theta_z = 0$, $d_z = 0$, and $W_{z,jk} = 1/[r(1 + C) + \varepsilon]$, for $1 \leq j \leq n$ and $1 \leq k \leq r$. Since ε is a small random number introduced only for symmetry breaking, we can ignore it in the following analysis. Thus, we let $W_{z,jk}$ be $1/[r(1 + C)]$. Let the active terminal be j' when z is activated for the first time, i.e., at this time $I_{j'} = V_{j'1} = 1$, and no other SR units have the activity level of 1. According to (1),

$$E_z = \sum_{j,k} W_{z,jk} V_{jk}(t) = \frac{1}{r(1 + C)} \sum_{j,k} V_{jk}(t)$$

Limited by the number of SR units in each shift-register assembly (the STM capacity) $\Sigma_{jk} V_{jk}(t) < rI_{j'} = r$. Thus

$$E_z < \frac{1}{1 + C} \tag{17}$$

On the other hand, since $d_z = 0$, $A_z = 1$, one shot learning by (4) ends up with $W_{z,j'1} = 1/(1 + C)$, and $W_{z,jk} = 0$, for $j \neq j'$ or $k \neq 1$. Also θ_z becomes $1/(1 + C)$ according to (6). Because of this weight distribution and the new threshold, the only chance for unit z to be selected is when terminal j' is active again, namely $V_{j'1} = 1$ ($V_{j'1}$ is binary according to (3)). When $V_{j'1} = 1$, $E_z = W_{z,j'1} V_{j'1} = 1/(1 + C)$. Therefore, it will exceed the new threshold. Q.E.D.

Proposition 3: Except for $d = 0$ (initial training), an activated detector of degree d has u nonzero weights which have the distribution, $[1 - (u - i)\delta]/[C + E(u)]$, *for* $i = 1, \cdots, u$, where $u \leq d$. The activity and the threshold of the unit are both equal to $E(u)$.

Proof: We prove the proposition by mathematical induction on d. According to proposition 2, it is true for $d = 1$. Assume that this proposition is true for $d = \mathrm{q} \geq 1$. For $d = q + 1$, let us examine the time step when d increases from q to $q + 1$ in the next step. Again let the unit in question be z. Let v be the number of non-zero weights of z in this step. According to the inductive hypothesis, $E = \theta_z = h(v)$, $v \leq q$. Examine now time step t, the first time unit z is activated after d increases to $q+ 1$. Since the degree changes when a mismatch occurs in the next step (8), thus at t, the weight distribution and θ_z have not been changed. In order to exceed $\theta_z = h(v)$ with the weight distribution, the SR units corresponding to the non-zero weights must all have non-zero activities. According to the Cauchy-Schwartz inequality, the unit corresponding to the weight $[1 - (v - i)\delta]/[C + E(v)]$ must have the activity $1 - (v - i)\delta$. Let the sequence corresponding to these SR units be S: $p_1 - p_2 - \cdots - p_v$. Since $v \leq d$, $v \leq d+ 1$, thus the conclusion holds for this activation of z. Let us know study possible changes to the weight distribution, the activity and the threshold during this activation. There are two cases to be considered:

1) S is all that is currently held in STM. The weight update by (4) does not change the weight distribution, and the activity and the threshold of z will remain the same. Thus the conclusion holds for the next activation of z.
2) S is not all that is held in STM. Let S_a be the longest prior subsequence of S that is currently held in STM, satisfying $v < |S_a| + |S| = v' \leq q+ 1$, where $|X|$ denotes the length of the sequence X. Because the sensitivity parameter $A_z = 1 - (d-1)\delta$, one shot learning by (4) yields the weight distribution: $[1 - (v' - i)\delta]/[C + E(v')]$, for $i = 1, \cdots, v'$, corresponding to the sequence S_a-S. According to (6), the threshold is updated to $h(v')$. Since $h(v') > h(v)$, the threshold is increased as a result. Because of this new weight distribution and the new threshold, by the same token, unit z can be activated next time only by S_a-S, a left supersequence of S. Since $|S_a\text{-}S| \leq q+ 1$, the conclusion is true for that time.

The above two cases encompass all possible occasions when unit z is activated with its degree equal to $q+ 1$. Thus the proposition is true. Q.E.D.

We call that a detector is *triggered (activated)* by a sequence S, if the detector wins winner-take-all competition in the detector layer and S is the longest sequence whose components have a non-zero contribution (cf. (1)) to the activity of the detector. In other words, all the components of S have non-zero weights connecting to the detector. According to the above proposition and its proof, we readily have

Corollary 1: (a) The threshold update rule of (6) never decreases the threshold of an activated unit; (b) At any time, a detector can be triggered by only a single sequence; (c) Except for initial training, once a unit is activated by sequence S, it can only be activated by S or a left supersequence of S.

To be precise, (b) of the above corollary makes it legitimate to say that a detector is *tuned* to the unique sequence which can trigger a detector. This concept was used informally in Section III.B. We are now ready to prove the major conclusion of this paper, which is stated as the following theorem.

Theorem 1: A model defined in III.A with m detectors and r SR units for each of n shift-register assemblies can learn to produce an arbitrary sequence S of length $\leq m$ and degree $\leq r$.

Proof: Since learning is completed when there is no mismatch during an entire training sweep, in other words, the sequence anticipated by the model is the same as the sequence to be learned. Thus, when learning is completed, the model can produce the entire sequence with the presentation of the first component of S. So all that remains to be proven is that after a finite number of training sweeps, all mismatches are resolved. The following results need to be established first.

1°. *It is impossible that two detectors are tuned to the same sequence.* Once a detector i is tuned to a sequence S_i, it will be triggered by S_i when S_i occurs in the input unless a left supersequence of S_i is triggering another detector j. In the latter case, according to Proposition 1, detector j wins the competition. In any case, it is not possible to trigger another

detector and then adjust its weight distribution to make it tuned to S.

2°. *Once a detector is committed, it will anticipate a component.* If a detector i is activated, it is tuned to a subsequence S_i of S. Based on 1°, if S_i occurs again in the input it will trigger i unless a left supersequence S_j of S_i triggers another detector j. But there must exist an S_i in S that does not have a left supersequence to which another detector is already tuned. Otherwise, according to Proposition 1, i would not be activated in the first place. Therefore detector i will be activated at least once during the entire presentation of S. In other words, i will anticipate a component of S. When a detector is first committed, it is also activated, and the conclusion can be applied recursively when the sequence tuned by the detector is expanded as its degree increases. Therefore the conclusion is true.

We now prove that an arbitrary component p_i, except for the first one, of S can be uniquely anticipated after a finite number of sweeps. The first component needs to be provided in order to generate the sequence and thus is not anticipated by any detector. S can be written as $S_a - p_i - S_b$, for $i > 1$, and S_a is the prior subsequence of p_i from the beginning of S and S_b is the rest of S. When the entire S_a is presented, there will be a detector activated. If the detector fails to anticipate p_i, its degree increases. By 2°, there are at most $|S|-1$ committed detectors. Thus, unless p_i is uniquely anticipated by a detector, which validates the conclusion, the degree of the detector anticipating p_i will grow greater than or equal to the degree of p_i. By *unique* we mean the detector does not anticipate any other symbol except the one represented by p_i. Let the degree of p_i be d_i. We now examine an occasion when the degree of the detector currently anticipating p_i is greater than or equal to d_i. Again let the detector be unit z and d_z denote the degree of z. We have $d_z \geq d_i$. There are two cases to be considered:

1) $d_i \leq |S_a| = i-1$. In this case, after z is activated, z will be tuned to the context of p_i if $d_z = d_i \leq i-1$, or a left supersequence of the context of p_i. In any event, by the definition of a context, unit z will uniquely anticipate p_i.
2) $d_i > |S_a|$, as component D in the example sequence *A-B-D-A-B-E*. According to Proposition 3 and particularly the analysis of case 2) in the proof of the proposition, $\theta_z = h(i-1)$. Since $d_i > i-1$, S_a occurs as another part of S (see the above example). With that taken into account, S can be written as $S_c - S_a - p_j - S_d$, where p_j is a component, S_c and S_d are subsequences of S, and $|S_c| > 0$. The degree of the detector anticipating p_j will grow greater than $i-1$. Let the detector be z_j. Because $|S_c| > 0$, the threshold of z_j grows greater than $h(i-1)$. Thus, when S_a occurs as a beginning sequence, z_j will not be activated. The same reasoning applies to all the occurrences of S_a except for the one at the beginning of S. Therefore after finite number of sweeps, a detector will uniquely anticipate p_i. Q.E.D.

V. Other Model Properties

In this section, we point out some other properties of the model. Different from the previous section, the properties are presented through rather informal analysis and computer simulations.

A. Efficiency of Training

The number of training sweeps the model takes to learn a sequence largely depends on how complex the sequence is. One sweep is sufficient for learning a simple sequence, while more sweeps are needed for complex sequences (see the example in Section III.C). During each sweep, the detectors which do not correctly anticipate their next components have their degrees increased by at least one. A degree parameter increases more than once if the corresponding detector is triggered more than once (see the example in Section III.C). Let us know derive an upper bound on the number of training sweeps for a sequence of degree k. The following is the worst imaginable scenario. The detector which anticipates the component of degree k takes k sweeps to increase its degree to a value greater than or equal to k. Let the component be p_i. After that the detector will be able to uniquely anticipate the component. Furthermore, $k-1$ more sweeps are taken to increase the degree of a unit which anticipates a component whose context is the next right subsequence of the context of p_i, so that the unit can uniquely anticipate the component. Similar reasoning applies subsequently. So the total number of sweeps is $k+(k-1)+\cdots+1 = k(k+1)/2$. Since other less complex components are handled as well during these training sweeps, we conclude that an upper bound μ of the training algorithm satisfying,

$$\mu = \frac{k(k+1)}{2} \tag{18}$$

The above upper bound is not a tight one. We conjecture that a realistic estimate on average training cycles should be about k.

What happens if the degree of the sequence to be learned is greater than r, the number of SR units in a shift-register assembly? Due to insufficient capacity of STM, mismatches cannot be resolved based on what is held in STM, and the training process will not terminate. To prevent it from happening, a maximum number of training sweeps can be set based on the value of μ. While the degree of the sequence may not be available before training, r is a parameter of the network model and can be used to break endless training. Based on (18), training should be terminated once the number of training sweeps exceeds $r(r+1)/2$. We know for sure that a successful training does not take more than $r(r+1)/2$ sweeps.

In case of insufficient detector units, what will happen is that the network cannot find any detector whose activity level based on the current input passes its threshold. In other words, all detectors have value 0 as their activity level (see (1)) in response to the current input. In this case, the competitive mechanism should not select a unit arbitrarily. This requirement can be easily implemented by excluding a unit from competition if its activity is zero.

While a detector in Wang and Arbib [41] never increases its degree greater than the degree of the context it is supposed to detect, a detector in the present model may increase its

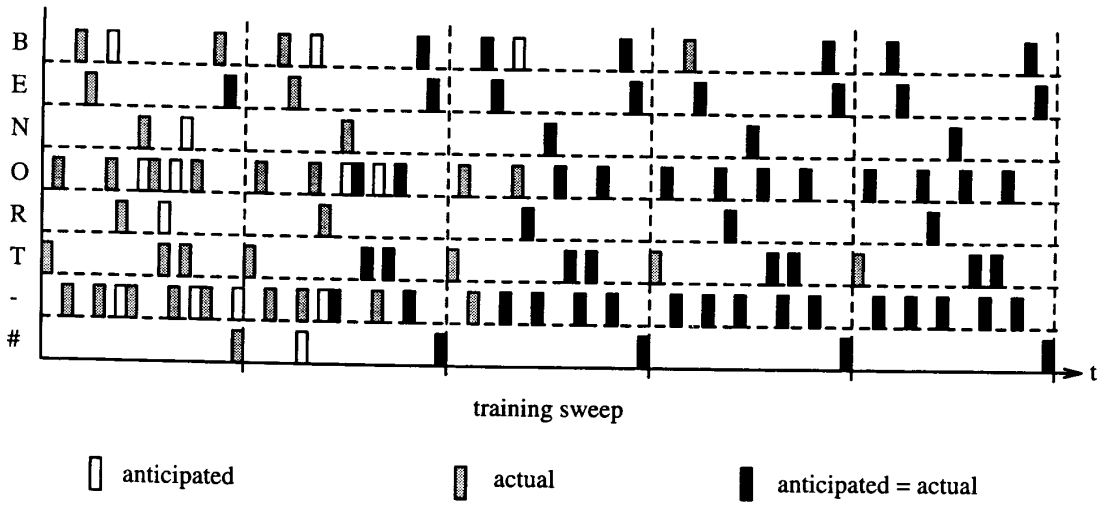


Fig. 5. Training and generation of the sequence ⟨TO-BE-OR-NOT-TO-BE⟩. Shown in the figure are the activity traces of the input terminals of a simulated network. See the legend of Fig. 4 for similar notations. The training phase took 5 training sweeps. The parameters used in the simulation are the same as listed in the caption of Fig. 4. The generation process can be initiated by presenting to the network the first component of the sequence, T.

degree unnecessarily to a value greater than the degree of the component it anticipates. In the example discussed in Section III.C, after training is completed the degree of detector 2 which anticipates component C is 4, while the degree of C in the sequence is 3. The reason for this to happen is that sometimes the degree of a detector is greater than the length of the subsequence that the detector is tuned to (proposition 3). When the degree of a detector increases due to a mismatch, the sequence that the detector is tuned to does not immediately get expanded accordingly. Instead, the detector has to wait until it is triggered again so that it has a chance to adjust its weight distribution. But the weight distribution will be adjusted based on the items held in STM. If the number of items held in STM is smaller than the degree of the detector, as analyzed in case 2) of the proof of proposition 3, the detector will be tuned to a sequence whose length is smaller than the degree of the detector. This situation is possible only for beginning components in a sequence. This is also noted in case 1) ° of the proof of theorem 1. However, as analyzed there, this situation will disappear shortly when the detector is triggered later in the presentation of the sequence. The worst situation probably occurs in the sequence A-B-A-C, where the degree of the detector anticipating C finally increases to 4, while the degree of C is 2. But in this case only two sweeps of training are sufficient to learn the sequence.

In terms of the number of detectors that are committed to learn a sequence, the present model has a significant advantage over the previous model by Wang and Arbib [41]. In their model, the number of detectors needed is equal to the length of the sequence minus one. In our model, no unit will be committed unless required. In this sense, it represents the most efficient use of units in the detector layer. Interestingly, this advantage is better manifested for complex sequences. In a rather extreme case, only two detectors need to be committed for the sequence A-A-$\cdots$-A-B, where B is preceded by k A's. One detector anticipates A with input A and another one anticipates B with the input of the sequence of kA's. Note that degree of the sequence is k. The network takes $\lceil k + 1/2 \rceil$ (ceiling function) sweeps to learn the sequence.

B. Computer Simulation

To further illustrate the model, we carried out the following computer simulation. The network has 24 detector units, each of which is associated with one modulator unit, 24 terminals each of which is connected to one shift-register assembly, and 6 SR units for each shift-register assembly (144 register units in total). See the figure legend for the other parameter values used in the simulation. Fig. 5 shows the activity trace of the network from a simulation run with input sequence ⟨*TO-BE-OR-NOT-TO-BE*⟩. As in the example of Section III.C, we use symbol '#' as the end marker. Also, symbol '-' here is treated as a distinct symbol separating meaningful words instead of a component separator. The behavior of the network is displayed in the same way as in Fig. 4. The network learned the sequence in 5 training sweeps. In the last training sweep, the system correctly anticipated every component of the sequence, as shown in the last column of the figure. After this training, the entire sequence can be correctly generated by the presentation of its first component, T in this case, and the activity trace will be the same as the last sweep of training. The degree of the sequence is 6, equal to r.

Once one sequence is learned, a right subsequence of it can be reproduced from a middle point of the sequence. In the above example, with symbol R as the initial input the network correctly produced the remaining part of the sequence ⟨*-NOT-TO-BE*⟩. Component R was chosen because it forms the degree 1 context for its successor '-'. In general, it requires a subsequence as an input to produce the remaining part of the sequence. The subsequence can activate an detector which anticipates a certain component. The component can then join the input to activate another detector, and so on, until the remaining part is fully generated. This feature of the model

conforms with the experience that one can often pick up a familiar song or a piece of music once exposed to a part of it.

C. Multiple Sequences

The network is capable of storing multiple input sequences. Multiple sequences can be learned either simultaneously or sequentially. In simultaneous training, all input sequences are presented one after another during each training sweep. After a number of sweeps, all sequences will be acquired by the network. As an example, we tested the network with three sequences N-E-U-R-A-L, M-A-C-H-I-N-E, and S-Y-S-T-E-M. After 6 training sweeps, all of them were able to be generated by the input of their first component (N, M, or S) respectively. In learning multiple sequences, each sequence is assumed to have a unique first component. The first component can be viewed as the identifier (name) of the sequence, which upon stimulation is able to produce the entire sequence. Once a sequence is acquired, its generation does not have to start from its identifier so long as a sufficient subsequence occurs in the input, the point explained in the previous subsection. For instance, after two sequences S_1-A-B-C-D and S_2-A-C-D-E are acquired, the first sequence S_1 (also the first component) can be either generated by the presentation of component S_1 or subsequence A-B. Similarly, after a human learner has learned a song, he or she can sing it either by being told the name of the song or by being primed with a beginning segment of the song.

Multiple sequences can also be trained sequentially, meaning that new ones can be learned after some sequences have been stored into the network. If a new sequence to be learned has no component in common with the stored ones, it can be trained and stored as if nothing were already stored in the network. The more interesting situation is that a new sequence has subsequences which also occur in the stored ones. In this case, as should be expected from the learning algorithm, some previous links between modulators and input terminals will be altered. For instance, assume that the simple sequence A-B-C was stored first, and now the network is being trained with another simple sequence D-B-E. The link from B to C will be replaced by a link from D-B to E as a result of learning the second sequence. In this sense, the previous memory is interfered as a result of learning new sequences. This effect, however, conforms with a well-known psychological phenomenon, and it is called *retroactive interference* [35]. The critical question is whether the interference can be overcome with a little retraining or not. In our model, the answer is yes. The committed detectors which are interfered are only those which have been tuned to a subsequence that occurs in the new sequence. With a little retraining, the degrees of appropriate detectors will be increased to differentiate the interfered old subsequences and the new one.

To demonstrate a typical situation of retroactive interference, the same network simulated in Section III.C was used to learn sequentially two sequences, S_3: R-E-M-E-M-B-E-R and S_4: M-E-M-O-R-Y. S_3 was trained first, and the network took 4 sweeps to learn it, as shown in Fig. 6(a). After that, S_4 was presented for training, and the network also took 4 sweeps to learn it (Fig.6(b)). Because of common subsequences in the two sequences, acquisition of S_4 interfered with the memory of S_3. The network needs to be retrained in order to correctly generate S_3. In the simulation, the network took two more sweeps to regain S_3, as shown in Fig. 6(c). After this relatively brief retraining to overcome retroactive interference, the network stored both sequences with no interference between them. Fig. 6(d) and (e) show two episodes of generating S_4 and S_3 with the input of their beginning components, respectively.

From the above discussion, it is clear that our model does not suffer from the so called *catastrophic interference* as exhibited in backpropagation learning of multilayer perceptrons [27], [34]. Catastrophic interference refers to the behavior where training of new associations between inputs and outputs destroys previously learned pattern associations. On the contrary, the present model exhibits the kind of retroactive interference similar to the limitations that people have in sequential learning. We call this behavior of sequential training *incremental learning*. Based on this result, we suggest that the conclusion drawn from the study of McCloskey and Cohen [27] and Ratcliff [34] may be applicable only to multilayer perceptrons with backpropagation training, not artificial neural networks in general.

VI. General Discussion

As evident in Theorem 1, the capacity r of the STM model limits the degree of a sequence that can be learned. Although r can be freely chosen in engineering applications, psychological evidence suggests that human STM can hold only a limited number of items, about seven [30]. Given such severe limitations, how can the model learn sequences with much longer repeating sequences? We suggest to use chunking to solve the problem. It was previously demonstrated that chunking provides an effective way of recognizing long sequences [41]. The basic idea there is to introduce hierarchies so that units in a particular hierarchy recognize a sequence of limited items of the next lower level. So in terms of sequence components which correspond to the lowest level, a unit in a certain level can recognize sequences of lengths exponentially increasing with the level of the hierarchy where the unit lies. Thus the mechanism significantly expands the lengths of the sequences that can be recognized. We think that a similar idea can be applied to this model of generation to increase the length of the context recognized by a specific detector. Instead of having one layer of detectors, multiple layers should be included in the model. Future study will need to address questions arising from the introduction of multiple layers, such as which level a detector should be selected from and how learning is performed within multiple layers if a unit in a higher layer is chosen.

Although our discussion so far focuses on temporal sequence generation, the model with some straightforward revision can also serve for sequence recognition. This is because the model has a component of context recognition which can be extended to arbitrary sequence recognition. In this situation, competitive learning in the detector layer should be

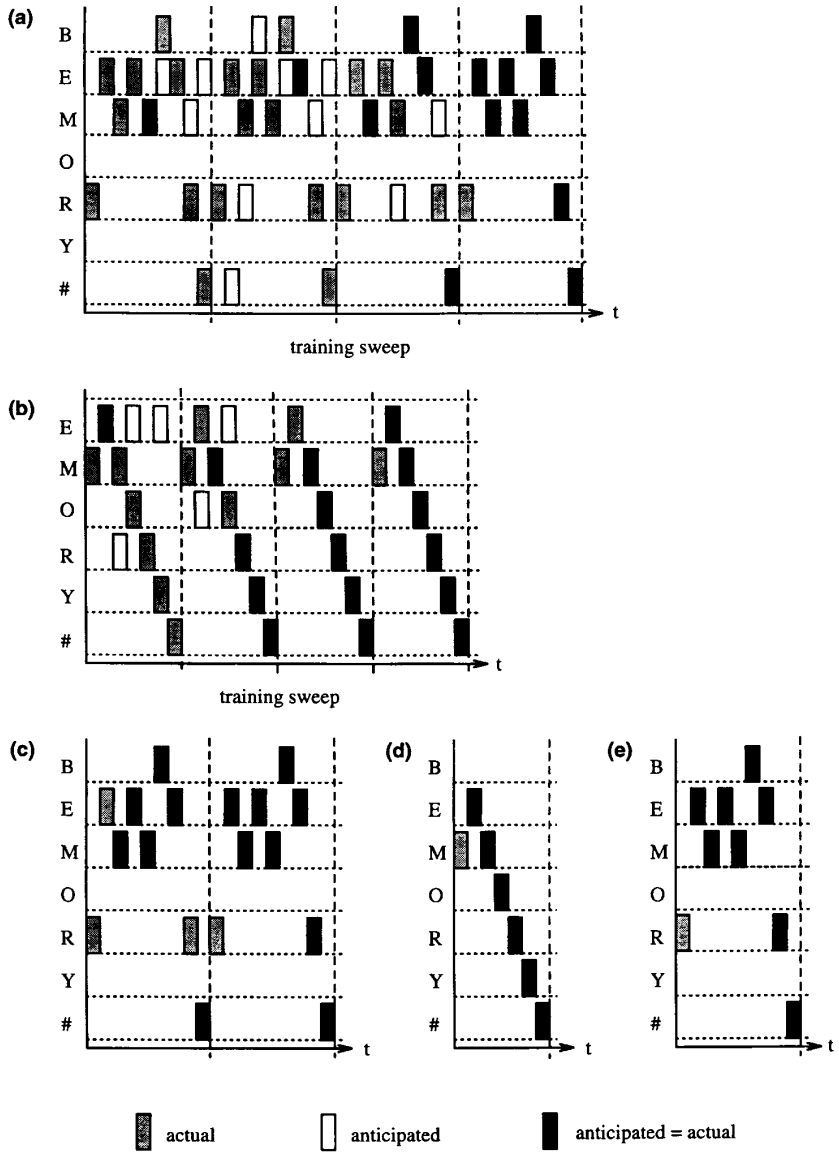


Fig. 6. Sequential training of two temporal sequences. The figure notations are the same as in Fig. 4. (a) Training of the network to generate sequence S_3: R-E-M-E-M-B-E-R. The network took four sweeps to learn the sequence. (b) Training of the same network to generate sequence S_4: M-E-M-O-R-Y. The network took four sweeps to learn the sequence. (c) The network was retrained to generate S_3. Two more training sweeps were required in order to correctly generate S_3. (d) The network correctly generated S_4 without further training. (e) The network correctly generated S_3 without further training. The parameter values used are the same as in Fig. 4.

triggered by some segmentation mechanism signaling the end of a sequence. The temporal masking mechanism leads to the desirable property that a detector tuned to a sequence will win the competition over those detectors tuned to the sequences which are right subsequences of the sequence. Acquisition and recognition of temporal sequences can be integrated in the same model. A recent model by Gjerdingen [12] demonstrates certain properties of temporal masking. His model embeds a masking field of Cohen and Grossberg [8] in an ART 3 network proposed by Carpenter and Grossberg [6] for recognizing temporal sequences of musical chords. It is not clear from the model description, however, how learning and recognition are combined into a single process of self-organization.

In this model, we have demonstrated how anticipation may be used to learn to generate complex temporal behaviors. The idea of anticipation-based learning seems to be consistent with psychological evidence about human learning of sequential behaviors. Meyer [29] proposed that expectation is key to music cognition. Nissen and Bullemer [33] reported that when a temporal sequence is repeatedly presented to subjects, their reaction to a particular component of the sequence becomes faster and faster, and the reaction time to a component in

a repeated sequence is much shorter than when it occurs in random sequences. The latter rules out the possibility that the reduction in reaction time is due to the familiarity with a component. This basic finding has been confirmed by later experiments [42], [7]. The result suggests that the subjects have developed with practice some form of anticipation before a particular component actually occurs in the sequence. It is observed that in learning temporal sequences human subjects can even be explicitly aware of the temporal structure of a sequence, and predict what comes next in the sequence [33], [42], [4].

One architectural characteristic of this model is the use of shift registers for maintaining a signal for a short time (Fig. 1). Similar architectures have been used by others for temporal pattern processing (see [37], [38]). This architecture was argued to be neurally plausible [20]. In the auditory cortex of cats, electrophysiological recordings identify various time delays up to 1.6 seconds in response to the same tones separated by certain periods or a sequence of different tones [19], [28]. In the visual system, Anderson and van Essen [1] have argued that shift circuits exist at many levels in the visual pathway (so called the shifter circuit hypothesis), and they have discussed a range of computational functions of such shift circuits.

Along the similar lines taken by Wang and Arbib [41], the present model argues from the computational perspective for the chaining theory of temporal behavior as rejected by Lashley [24], echoing more recent psychological theories of serial order organization [32], [25]. Simple associative chaining between adjacent sequence components is unlikely to be true. However, as demonstrated in this paper, if chaining between remote components and chunking of subsequences into high-order components are allowed, much more complex temporal behaviors can be realized with the basic idea of associative chaining, going beyond what was discussed by Lashley [24]. The present model shows how learning and generation of complex temporal sequences can be computationally achieved by self-organizing a neural network. We also realize that chunking can significantly enhance the abilities of neural networks for temporal processing.

How self-organization of chunking is achieved in a neural network model has been hardly addressed at all, and it remains to be a challenge for the future study of temporal pattern processing. The present model encodes symbols of a temporal sequence directly by units in a network. This high-level encoding scheme is unlikely to be true in terms of brain mechanisms, whereby it is widely accepted that high-level symbols should be represented by some distributed activities across a population of cells. Because symbols are represented by individual units, a number of issues cannot be addressed, such as robustness of the network with respect to malfunctioning of network units and invariant recognition of individual symbols. Also left unanswered is how to detect regularities in a set of sequences, like syntax formation from a set of exemplar sequences.

Let us conclude that in this paper a neural network model is presented which learns to generate complex temporal sequences. Sequences are acquired by one-shot learning obeying a normalized Hebbian learning rule, in combination with a competitive mechanism realized by a winner-take-all network. During learning and generation, the network actively anticipates the next component on the basis of a previously stored context. A mismatch between system anticipation and actual input triggers self-organization of context expansion. Our formal analysis has demonstrated the functioning of the network and shown that the model can learn to generate any complex sequences within a certain limit determined by the network architecture. In addition, multiple sequences can be acquired in an incremental fashion. Finally, several aspects of the model are discussed with respect to neurobiological and psychological studies of temporal pattern processing.

REFERENCES

[1] C. H. Anderson and D. C. van Essen, "Shifter circuits: A computational strategy for dynamic aspects of visual processing," *Proc. Natl. Acad. Sci. USA*, vol. 84, pp. 6297–6301, 1987.

[2] G. Bradski, G. A. Carpenter, and S. Grossberg, "Working memory networks for learning temporal order with application to three-dimensional visual object recognition," *Neural Computat.*, vol. 4, pp. 270–286, 1992.

[3] J. Buhmann and K. Schulten, "Noise-driven temporal association in neural networks," *Europhys. Lett.*, vol. 4, pp. 1205–1209, 1987.

[4] T. Curran and S. W. Keele, "Attentional and nonattentional forms of sequence learning," *J. Exp. Psychol: Learning, Memory, and Cognition*, vol. 19, pp. 189–202, 1993.

[5] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vision, Graphics, Image Process.*, vol. 37, pp. 54–115, 1987.

[6] ——, "ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architecture," *Neural Networks*, vol. 3, pp. 129–152, 1990.

[7] A. Cohen, R. I. Ivry, and S. W. Keele, "Attention and structure in sequence learning," *J. Exp. Psychol.*, vol. 16, pp. 17–30, 1990.

[8] M. A. Cohen and S. Grossberg, "Masking fields: A massively parallel neural architecture for learning, recognizing, and predicting multiple groupings of patterned data," *Appl. Opt.*, vol. 26, pp. 1866–1891, 1987.

[9] T. Dehaene, J. P. Changeux, and J. P. Nadal, "Neural networks that learn temporal sequences by selection," *Proc. Natl. Acad. Sci. USA*, vol. 84, pp. 2727–2731, 1987.

[10] K. Doya and S. Yoshizawa, "Adaptive neural oscillator using continuous-time back-propagation learning," *Neural Networks*, vol. 2, pp. 375–385, 1989.

[11] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, pp. 179–211, 1990.

[12] R. O. Gjerdingen, "Learning syntactically significant temporal patterns of chords: A masking field embedded in an ART 3 architecture," *Neural Networks*, vol. 5, pp. 551–564, 1992.

[13] S. Grossberg, "Some networks that can learn, remember, and reproduce any number of complicated space-time patterns, I," *J. Math. Mechan.*, vol. 19, pp. 53–91, 1969.

[14] ——, "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors," *Biol. Cybern.*, vol. 23, pp. 121–134, 1976.

[15] I. Guyon, L. Personnaz, J. P. Nadal, and G. Dreyfus, "Storage and retrieval of complex sequences in neural networks," *Phys. Rev. A*, vol. 38, pp. 6365–6372, 1988.

[16] M. J. Healy, T. P. Caudell, and S. D. G. Smith, "A neural architecture for pattern sequence verification through inferencing," *IEEE Trans. Neural Networks*, vol. 4, pp. 9–20, 1993.

[17] D. O. Hebb, *The organization of behavior.* New York: Wiley, 1949.

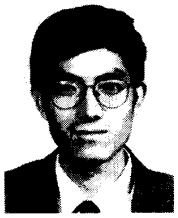
[18] T. M. Heskes and S. Gielen, "Retrieval of pattern sequences at variable speeds in a neural network with delays," *Neural Networks*, vol. 5, pp. 145–152, 1992.

[19] S. Hocherman and E. Gilat, "Dependence of auditory cortex evoked unit activity on interstimulus interval in the cat," *J. Neurophysiol.*, vol. 45, pp. 987–997, 1981.

[20] J. J. Hopfield and D. W. Tank, "Neural architecture and biophysics for sequence recognition," in *Neural Models of Plasticity*, J. H. Byrne and W. O. Berry, Eds. San Diego, CA: Academic Press, 1989.

[21] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Proc. 8th Ann. Conf. Cognit. Sci. Soc.* Hillsdale, NJ: Erlbaum, 1986, pp. 531–546.

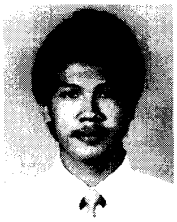
[22] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, pp. 1464–1480, 1990.
[23] B. Kosko, "Bidirectional associative memory," *IEEE Trans. Syst. Man Cyber.*, vol. SMC-18, pp. 49–60, 1988.
[24] K. S. Lashley, "The problem of serial order in behavior," in *Cerebral Mechanisms in Behavior*, L.A. Jeffress, Ed. New York: Wiley & Sons, pp. 112–146, 1951.
[25] S. Lewandowsky and B. B. Murdock, Jr., "Memory for serial order," *Psychol. Rev.*, vol. 96, pp. 25–57, 1989.
[26] C. v. d. Malsburg, "Self-organization of orientation sensitive cells in the striate cortex," *Kybernetik*, vol. 14, pp. 85–100, 1973.
[27] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychol. of Learning and Motivat.*, vol. 24, pp. 109–165, 1989.
[28] T. M. McKenna, N. M. Weinberger, and D. M. Diamond, "Responses of single auditory cortical neurons to tone sequences," *Brain Res.*, vol. 481, pp. 142–153, 1989.
[29] L. B. Meyer, *Emotion and meaning in music*. Chicago, IL: University of Chicago Press, 1956.
[30] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychol. Rev.*, vol. 63, pp. 81–97, 1956.
[31] M. C. Mozer, "Neural net architectures for temporal sequence processing," in *Predicting the Future and Understanding the Past*, A. Weigend and N. Gershenfeld, Eds. Redwood City, CA: Addison-Wesley, 1993.
[32] B. B. Murdock, Jr., "Serial-order effects in a distributed-memory model," in *Memory and Learning: The Ebbinghaus Centennial Conference*, D. S. Gorfein and R. R. Hoffman, Eds. Hillsdale, NJ: Erlbaum, pp. 227–310, 1987.
[33] M. J. Nissen and P. Bullemer, "Attentional requirements of learning: Evidence from performance measures," *Cognit. Psychol.*, vol. 19, pp. 1–32, 1987.
[34] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting function," *Psychol. Rev.*, vol. 97, pp. 285–308, 1990.
[35] S. K. Reed, *Cognition: Theory and Applications*. Monterey, CA: Brooks/Cole, 1982.
[36] H. Sompolinsky and I. Kanter, "Temporal association in asymmetric neural networks," *Phys. Rev. Lett.*, vol. 57, pp. 2861–2864, 1986.
[37] D. W. Tank and J. J. Hopfield, "Neural computation by concentrating information in time," *Proc. Natl. Acad. Sci. USA*, vol. 84, pp. 1896–1900, 1987.
[38] A. Waibel, T. Hanazawa, G. E. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust. Speech and Signal Process.*, vol. 37, pp. 328–339, 1989.
[39] D. L. Wang, "Temporal pattern processing," in *Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995, in press.
[40] D. L. Wang and M. A. Arbib, "Complex temporal sequence learning based on short-term memory," *Proc. IEEE*, vol. 78, pp. 1536–1543, 1990.
[41] ——, "Timing and chunking in processing temporal order," *IEEE Trans. Syst. Man Cyber.*, vol. 23, no. 4, pp. 993–1009, 1993.
[42] D. B. Willingham, M. J. Nissen, and P. Bullemer, "On the development of procedural knowledge," *J. Exp. Psychol.: Learning, Memory, and Cognition*, vol. 15, pp. 1047–1060, 1989.



DeLiang Wang (M'90) was born in Anhui, the People's Republic of China, in 1963. He received the B.Sc. degree in 1983 and the M.Sc. degree in 1986 from Beijing University, Beijing, China, and the Ph.D. degree in 1991 from the University of Southern California, Los Angeles, CA, all in computer science.

From July 1986 to December 1987, he was with the Institute of Computing Technology, Academia Sinica, Beijing. He is currently an Assistant Professor of Department of Computer and Information Science and Center for Cognitive Science at The Ohio State University, Columbus, OH.

His present research interests include temporal pattern processing, auditory and visual pattern perception, neural network theories, and computational neuroscience. He is a member of IEEE Computer Society, IEEE Systems, Man, and Cybernetics Society, and the International Neural Network Society and AAAS.



Budi Yuwono received the B.Sc. degree in Industrial Engineering from the Bandung Institute of Technology, Bandung, Indonesia in 1986, the M.Sc. degree in computer science from The Ohio State University, Columbus, OH, in 1991.

During 1987–1988, he was a research and education staff member at the Inter University Center for Computer Science at the University of Indonesia, Jakarta, Indonesia. He is currently a Ph.D. candidate at the Department of Computer and Information Science, The Ohio State University.

His research interests are in information retrieval, artificial intelligence, machine learning, and parallel and distributed computing.