In R. Sun and C.L. Giles (Eds.): *Sequence Learning*, LNAI 1828, pp. 53-79, 2000. Springer-Verlag, Berlin Heidelberg.

Anticipation Model for Sequential Learning of Complex Sequences*

DeLiang Wang

Department of Computer and Information Science
and Center for Cognitive Science
The Ohio State University

1 Introduction

One of the fundamental aspects of human intelligence is the ability to process temporal information (Lashley, 1951). Learning and reproducing temporal sequences are closely associated with our ability to perceive and generate body movements, speech and language, music, etc. A considerable body of neural network literature is devoted to temporal pattern generation (see Wang, 2001, for a recent review). These models generally treat a temporal pattern as a sequence of discrete patterns, called a temporal sequence. Most of the models are based on either multilayer perceptrons with backpropagation training or the Hopfield model of associative recall. The basic idea for the former class of models is to view a temporal sequence as a set of associations between consecutive components, and learn these associations as input-output transformations (Jordan, 1986; Elman, 1990; Mozer, 1993). To deal with temporal dependencies beyond consecutive components, part of the input layer is used to keep a trace of history, behaving as short-term memory (STM). Similarly, for temporal recall based on the Hopfield associative memory, a temporal sequence is viewed as associations between consecutive components. These associations are stored in extended versions of the Hopfield model that includes some time delays (Sompolinsky & Kanter, 1986; Buhmann & Schulten, 1987; Heskes & Gielen, 1992). To deal with longer temporal dependencies, high-order networks have been proposed (Guyon et al., 1988).

1.1 Learning complex sequences

One of the main problems with the above two classes of models lies in the difficulty in retrieving complex temporal sequences, where the same part may occur many times in the sequence. Though proposed remedies can alleviate the problem to some degree, the problem is not completely resolved. In multilayer perceptrons, a blended form of STM becomes increasingly ambiguous when temporal dependencies increase (Bengio et al., 1994). The use of high-order units

* Thanks to X. Liu for his help in typesetting. The preparation of this chapter was supported in part by an ONR YIP award and a grant from NUWC.

in the Hopfield model requires a huge number of connections to deal with long range temporal dependencies, or the model yields ambiguities.

More recently, Bradski et al. (1994) proposed an STM model, which exhibits both recency and primacy, where the former means that more recent items in a sequence are better retained and the latter means that the beginning items are less prone to forgetting. Both recency and primacy are characteristics of human STM. In addition, their model creates new representations for repeated occurrences of the same symbol, thus capable of encoding complex sequences to a certain extent. Granger et al. (1994) proposed a biologically motivated model for encoding temporal sequences. Their model uses a competitive learning rule that eventually develops sequence detectors at the end of sequence presentation. Each detector encodes a sequence whereby the beginning component has the highest weight, and the subsequent components have successively lower weights. They claim that the network has an unusually high capacity. However, it is unclear how their network reads out the encoded sequences. Baram (1994) presented a model for memorizing vector sequences using the Kanerva memory model (Kanerva, 1988). The basic idea is similar to those models that are based on the Hopfield model. Baram's model uses second-order synapses to store the temporal associations between consecutive vectors in a sequence, but the model deals only with sequences that contain no repeating vectors. Rinkus (1995) proposed a model of temporal associative memory, based on associations among random sequences. The associations are encoded using a method similar to the associative memory of Willshaw et al. (1969). However, the model needs an additional operation that maps a sequence component in a semi-random vector and remembers the mapping for later decoding.

Based on the idea of using STM for resolving ambiguities, Wang and Arbib (1990) proposed a model for learning to recognize and generate complex sequences . With an STM model, a complex sequence is acquired by a learning rule that associates the activity distribution in STM with a context detector (for a rigorous definition see Section 2). For sequence generation, each component of a sequence is associated with a context detector that learns to associate with the component. After successful training, a beginning part of the sequence forms an adequate context for activating the next component, and the newly activated component joins STM to form a context for activating the following component. This process continues until the entire sequence is generated. A later version (Wang & Arbib, 1993) deals with the issues of time warping and chunking of subsequences. In particular, sequences in this version can be recognized in a hierarchical way and without being affected by presentation speed. Hierarchical recognition enables the system to recognize sequences whose temporal dependencies are much longer than the STM capacity. In sequence generation, the system is capable of maintaining relative timing among the components while the overall rate can change.

Recently, L. Wang (1999) proposed to use multi-associative neural networks for learning and retrieving spatiotemporal patterns. STM is coded by systematic delay lines. The basic idea is that, when dealing with complex sequences ,

one pattern is allowed to be associated with a set of subsequent patterns, and disambiguation can be eliminated by intersecting multiple sets associated by the previous pattern, the pattern prior to the previous pattern, and so on. It is easy to see that a complex sequence can be unambiguously generated with a sufficient number of systematic delay lines. Associations between spatial patterns are established through single units in a competitive layer. A major drawback of the multi-associative network model is that much of system architecture and many network operations are algorithmically described, rather than arising from an autonomous neural network (e.g., no discussion on how set intersection is neurally implemented).

1.2 Sequential Learning Problem

A comprehensive model of temporal sequence learning must address the issue of sequentially learning multiple sequences; that is, how are new sequences learned after some sequences have been acquired? One way of learning multiple sequences is to use simultanous training, where many sequences are learned at once. A model that can learn one sequence can generally be extended to learn multiple sequences with simultaneous training. A straightforward way is to concatenate multiple sequences into a single long sequence. Given that each sequence has a unique identifier, a model can learn all of the sequences if it can learn the concatenated sequence. However, sequential learning of multiple sequences is an entirely different matter. It is a more desirable form of training because it allows the model to acquire new sequences without bringing back all the previously used sequences - a form of *incremental learning* . Incremental learning not only conforms well with human learning, but also is important for many applications that do not keep all the training data and where learning is a long-term on-going process.

It turns out that incremental learning is a particularly challenging problem for neural networks. In multilayer perceptrons, it is well recognized that the network exhibits so called catastrophic interference, whereby later training disrupts the traces of previous training. It was pointed out by Grossberg (1987), and systematically revealed by McCloskey and Cohen (1989) and Ratcliff (1990). Many subsequent studies attempt to address the problem, and most of proposed remedies amount to reducing overlapping in hidden layer representations by some form of orthogonalization, a technique used long ago for reducing cross-talks in associative memories (see Kruschke, 1992, Sloman & Rumelhart, 1992, and French, 1994). Most of these proposals are verified only by small scale simulations, which, together with the lack of rigorous analysis, make it difficult to judge to what extent the proposed methods work. It remains to be seen whether a general remedy can be found for multilayer perceptions. Associative memories are less susceptible to the problem, and appear to be able to incorporate more patterns easily so long as the overall number of patterns does not exceed the memory capacity. However, the Hopfield model has a major difficulty in dealing with correlated patterns with overlapping components (Hertz et al., 1991). The Hopfield model has been extended to deal with correlated patterns (Kantor &

Sompolinsky, 1987), and Diederich and Opper (1987) proposed a local learning rule to acquire the necessary weights iteratively. The local learning rule used by them is very similar to the perceptron learning rule. Thus, it appears that such a scheme for dealing with correlated patterns would suffer from catastrophic interference.

The major cause of catastrophic interference is the *distributedness* of representations; the learning of new patterns needs to use and alter those weights that participate in representing previously learned patterns. There is a tradeoff between distributedness and interference. Models that use non-overlapping representations, or local representations, do not exhibit the problem. For example, the ART model (Carpenter & Grossberg, 1987) does not have the problem because each stored pattern uses a different weight vector and no overlapping is allowed between any two weight vectors.

During sequential learning , humans show some degree of interference. Retroactive interference has been well documented in psychology (Crooks & Stein, 1991), which occurs when learning a later event interferes with the recall of earlier information. In general, the similarity between the current event and memorized ones is largely responsible for retroactive interference (Barnes & Underwood, 1959; Chandler, 1993; Bower et al., 1994). Animals also exhibit retroactive interference (Rodriguez et al., 1993). The existence of retroactive interference suggests that events are not independently stored in the brain, and related events are somehow intertwined in the memory. Although recall performance of the interfered items decreases, it still is better than the chance level, and it is easier to relearn these items than to learn them for the first time. This analysis suggests that a memory model that stores every item independently cannot adequately model human/animal memory. From the computational perspective, the models that store different events in a shared way have a better storage efficiency than those that do not. In summary, a desired memory model should exhibit some degree of retroactive interference when learning similar events, but not catastrophic interference.

In this chapter, we describe the *anticipation model* for temporal sequence learning (Wang & Yuwono, 1995; Wang & Yuwono, 1996). Similar to Wang and Arbib (1990), an STM model is used for maintaining a temporal context. In learning a temporal sequence, the model actively anticipates the next component based on STM. When the anticipation is correct, the model does nothing and continues to learn the rest of the sequence. When the anticipation is incorrect, namely a mismatch occurs, the model automatically expands the context for the component. A one-shot (single step) normalized Hebbian learning rule is used to learn contexts, and it exhibits the mechanism of temporal masking, where a sequence masks its subsequences in winner-take-all competition. The anticipation model can learn to generate an arbitrary sequence by self-organization, thus avoiding supervised teaching signals as required in Wang and Arbib. Furthermore, the model is examined in terms of its performance on sequential training tasks. We show that the anticipation model is capable of incremental learning ,

and exhibits retroactive interference but not catastrophic interference. Extensive simulations reveal that the amount of retraining is relatively independent of the number of sequences stored in the model. Furthermore, a mechanism of chunking is described that creates chunks for recurring subsequences. This chunking mechanism significantly improves training and retraining performance.

The remaining part of the chapter is organized as follows. In Section 2, the anticipation model is fully defined. Section 3 introduces several rigorous results of the anticipation model. In Section 4, we provide simulation results of the model, in particular for learning many sequences incrementally. The simulation results suggest that incremental learning in the anticipation model is capacity-independent, or unaffected by the number of stored sequences. Section 5 describes the chunking mechanism and shows how chunking improves the learning performance. Section 6 provides some general discussions about the anticipation model. Finally, Section 7 concludes the chapter.

2 Anticipation Model

We follow the terminology introduced by Wang and Arbib (1990). Sequences are defined over a symbol set Γ, which consists of all possible symbols, or spatial (static) patterns. Sequence S of length N over Γ is defined as p_1-p_2-...-p_N, where $p_i (1 \leq i \leq N) \in \Gamma$ is called a component of S. The sequence p_j-p_{j+1}-...-p_k, where $1 \leq j \leq k \leq N$, is a *subsequence* of S, and the sequence p_j-p_{j+1}-...-p_N where $1 \leq j \leq N$, is a *right subsequence* of S. In general, in order to produce a component by its predecessors in a sequence, a prior subsequence is needed. For example, to produce the first "P" in the sequence *M-I-S-S-I-S-S-I-P-P-I* requires the prior subsequence *S-I-S-S-I*. This is because *I-S-S-I* is a recurring subsequence. Hence, the *context* of p_i is defined as the shortest prior subsequence of p_i that uniquely determines p_i in S. The *degree* of p_i is the length of its context. The degree of S is the maximum degree of all of the components of S. Therefore, a *simple sequence*, where each component is unique, is a degree 1 sequence and a *complex sequence*, which contains recurring subsequences, is a sequence whose degree is greater than 1.

2.1 Basic Network Description

We now describe the basic components of the anticipation model. Fig. 1 shows the architecture of the network. The network consists of a layer of n input terminals, each associated with a shift-register (SR) assembly, and a layer of m context detectors, each associated with a modulator. Each SR assembly contains r units, arranged so that the input signal stimulating an input terminal shifts to the next unit every time step. SR assemblies serve as STM for input signals. Each detector receives input from all SR units, and there are lateral connections, including self-excitation, within the detector layer that form winner-take-all architecture. These connections and competitive dynamics lead to the detector that receives the greatest ascending input from SR units to be the sole winner of the entire

detector layer. In addition to the ascending and lateral connections, each detector also connects mutually with its corresponding modulator, which in turn connects directly with input terminals.

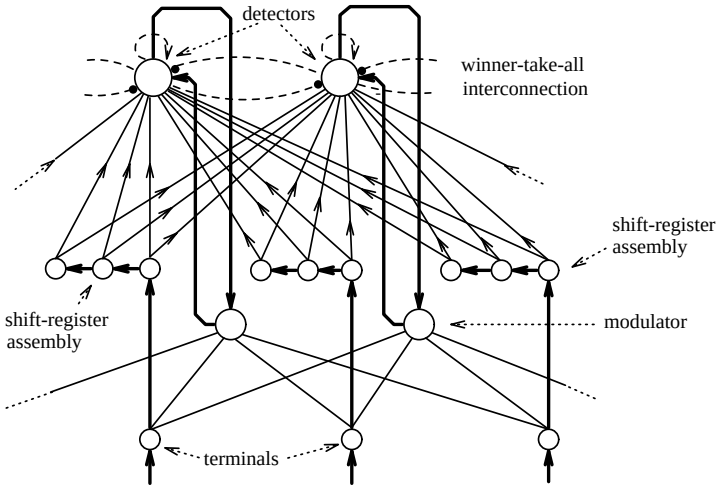

Fig. 1. Architecture of the anticipation model. Thin solid lines denote modifiable connections, and thick or dash lines denote fixed connections. The connections between terminals and modulators are bidirectional.

The model internally anticipates the next component and compares it with the external input through the modulator layer. 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 corresponding modulator in the next time step (assuming some delay). Once enabled, the modulator performs one-shot learning that 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 an enabled modulator. Basically, this one-shot learning establishes the association between a context detector and the next input component. If the active terminal and the enabled modulator do not match next time when the detector is activated, the anticipated activation of the modulator will be absent. This mismatch will be detected by the modulator, which in turn will send a signal to its respective detector to expand the context that the detector is supposed to recognize.

2.2 Model Description

The activity of detector i at time t, $E_i(t)$, is defined as:

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

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

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 winner-take-all competition in the detector layer, to be discussed later. A_i is defined later in (5). The activity $V_{jk}(t)$ is given as follows,

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

where $I_j(t)$ is the binary activity of terminal j, that is, $I_j(t) = 1$ 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 equal to 1 at t. 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 but 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 $V_{jr}(t+r-1) = max(0, 1-\delta(r-1))$. 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, $\delta < 1/(r-1)$.

As mentioned earlier, 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 Grossberg (1976). In such a competitive network, the activity of each detector evolves until the network reaches equilibrium, at which point 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. Therefore, we assume that each discrete time step is longer than the time needed for the winner-take-all mechanism to settle at an equilibrium.

The learning rule for each detector i is a Hebbian rule (Hebb, 1949) plus normalization to keep the overall weight a constant (von der Malsburg, 1973; Wang & Arbib, 1990), and it is denoted as a *normalized Hebbian rule*,

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

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

where α is a gain parameter or learning rate. A large α makes training fast. It is easy to see that very large α leads to approximate one-shot learning. 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, let $O_i(t)$ equal 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, and thus more connections of the detector can be modified according to (4a). Furthermore, the sensitivity parameter A_i is adaptive by itself:

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

where d_i, indicating the degree of detector i, is a non-negative integer, initialized to 0. δ is the decay parameter 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 one SR unit - a head unit - when $d_i = 0$ or 1. When d_i increases, more SR units are sensed. 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 described in Sect. 3. The connection weight $W_{i,jk}$ is initialized to $1/[r(1+C)+\epsilon]$, where ϵ is a small random number introduced to break symmety between the inputs of the detectors, which may cause problems for competitive dynamics.

Let unit z be the winner of the competition in the detector layer. As a result of the updated connection weights, the activity of unit z will change when the same input is presented in the future. More specifically, E_z is monotonically non-decreasing as learning takes place. This observation will be further discussed in the next section. The resulting, increased, activity in (1) is then used to update the threshold of unit z. This is generally described as:

$$\theta_i(t+1) = \theta_i(t) + O_i(E_i^*(t+1) - \theta_i(t)) \tag{6}$$

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

A modulator receives both a top-down connection from its corresponding detector and bottom-up connections from input terminals (Fig. 1). 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,

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

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

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

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

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

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

This one-shot learning 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. This training results in a one-to-one association between a modulator and a terminal.

We now explain under what condition a mismatch occurs, which leads to an increment of the degree of the winning detector. According to (7), a mismatch occurs when $O_i(t-1) = 1$ and $\sum_{j=1}^{n} R_{ij} I_j(t) = 0$. Since at any time, only one bottom-up weight of modulator i equals 1 and only one input terminal (I_j) is active, mismatch occurs when the non zero weight and the terminal with non zero input do not coincide. But one-shot learning of Eq. 9 establishes a non zero link only between a modulator and the next input terminal. Therefore, a mismatch occurs if the link between detector i and an input terminal established last time when detector i was activated does not coincide with the active input terminal this time (at time t). The bottom-up links of a modulator established between the modulator (or the corresponding detector) and the next input component are used for the modulator to anticipate the next component in sequence generation. Thus a mismatch corresponds to where the anticipated input symbol does not match with the actual input 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 involved detectors to 1. For complex sequences , though, the degree of the relevant detectors need further increase until no mismatch occurs.

The training is repeated each 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 during the last training sweep. In this case the network correctly anticipates the next component for the entire sequence. 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. In this case, an inactive global unit by the end of a sweep signals the end of the training phase.

3 Analytical Results

In this section, we summarize several analytical results on the anticipation model. These results are listed in the form of propositions without proofs, and the interested reader is referred to Wang and Yuwono (1995; 1996) for detailed proofs of these results.

*Proposition*1. The normalized Hebbian rule of (4) with the following choice of parameter C

$$C > \frac{\delta r(r-1)}{6}[1 + \frac{\delta + 2}{1 - \delta(r-1)}] \tag{10}$$

leads to a property called *temporal masking*: the detector of sequence S is preferred to the detectors of 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. This property is called temporal masking, following the term *masking fields* introduced by Cohen and Grossberg (Cohen & Grossberg, 1987), which state that larger spatial patterns are preferred to smaller ones when activating their corresponding detectors.

Inequality (10) tells us how to choose C based on the value of δ in order to ensure that the detector of a sequence masks the detectors of its left subsequences. The smaller is δ, the smaller is the right-hand-side of (10), and thus the smaller C can be chosen to satisfy the inequality. As a degenerate case, if $\delta = 0$, (10) becomes $C > 0$, and this corresponds to exactly the condition of forming masking fields in static pattern recognition (Cohen & Grossberg, 1987). Therefore, (10) 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 SR units for STM (see the discussion in Sect. 2.2), thus the degree of the learnable sequences (see Eq. 3).

*Proposition*2. The following two conclusions result from the learning algorithm: (a) At any time, a detector can be triggered by only a single sequence; (b) Except for initial training, once a unit is activated by sequence S, it can only be

activated by S or a sequence that has S as a right subsequence. Because of (a), one can say that a detector is *tuned* to the unique sequence which can trigger a detector.

*Proposition*3. An anticipation model with m detectors, and r SR units for each of n SR assemblies can learn to generate an arbitrary sequence S of length $\leq m$ and degree $\leq r$, where S is composed of symbols from Γ with $|\Gamma| \leq n$.

Once training is completed, the network can be used to generate the sequence it has been trained on. During sequence generation, the learned connections from input terminals to modulators are used reversely for producing input components. Sequence generation is triggered by the presentation of the first component, or a sequence identifier. This presentation will be able to trigger an appropriate detector which then, through its modulator, leads to the activation of the second component. In turn, the newly activated terminal adds to STM, which then forms an appropriate context to generate another component in the sequence. This process continues until the entire sequence is generated.

Aside from Proposition 3, learning is efficient - it generally takes just a few training sweeps to acquire a sequence. This is because the anticipation model employs the strategy of least commitment. The model views, as a default, the sequence to be learned as a simple one, and expands the contexts of sequence components only when necessary. Another feature of the model is that, depending on the nature of the sequence, the system can yield significant sharing among context detectors: the same detector may be used for anticipating the same symbol that occurs many times in a sequence. As a result, the system needs fewer detectors to learn complex sequences than the model of Wang and Arbib (1990; 1993).

The above results are about learning a single sequence, whereby a sequence is presented to the network one component at a time during training. When dealing with multiple sequences, each sequence is assumed to be unique, because learning a sequence that has been acquired corresponds to recalling the sequence. We assume that the first component of a sequence represents the unique identifier of the sequence. To facilitate the following exposition, we define a *sequential learning procedure* as the following. The training process proceeds in rounds. In the first round, the first sequence is presented to the network in repeated sweeps until the network has learned the sequence. The second round starts with the presentation of the second sequence. Once the second sequence is acquired by the network, the network is checked to see if it can generate the first sequence correctly when presented with the identifier of the sequence. If the network can generate the first sequence, the second round ends. Otherwise, the first sequence is brought back for retraining. In this case, the first sequence is said to be *interfered* by the acquisition of the second sequence. If the first sequence needs to be retrained, the second sequence needs to be checked again after the retraining of the first sequence is completed, since the latter lead to the interference of the second sequence. The second round completes when both sequences can be produced by the network. In the third round, the third sequence is presented to

the network repeatedly until it has been learned. The network is then checked to see if it can generate the first two sequences; if yes, the third round is completed; if not, retraining is conducted. In the latter case, retraining is always conducted on the sequences that are interfered. The system sequentially checks and retrains each sequence until every one of the three sequences can be generated by the network - that ends the third round. Later sequences are sequentially trained in the same manner. It is possible that a sequence that is not interfered when acquiring the latest sequence gets interfered as a result of the retraining of some other interfered sequences. Because of this, retraining is conducted in a systematic fashion as the following. All of the previous sequences plus the current one are checked sequentially and retrained if interfered. This retraining process is conducted repeatedly until no more interference occurs for every sequence learned so far. Each such process is called a *retraining cycle*. Thus a round in general consists of repeated retraining cycles.

If a system exhibits catastrophic interference, it cannot successfully complete a sequential learning procedure with multiple sequences. The system instead will show endless *oscillations* between learning and relearning different sequences. In the case of two sequences, for example, the system can only acquire one sequence - the latest one used in a sequential training procedure. Thus, the system will be stuck in the second round.

*Proposition*4. Given sufficient numbers of detectors and SR units for each shift-register assembly, the anticipation model can learn to produce a finite number of sequences sequentially.

In Proposition 4, the number of units in each SR assembly, or the STM capacity, must be sufficient to handle long temporal dependencies in the context of multiple sequences. It should be clear that the complexity of a sequence may increase when it is trained with other sequences. For example, X-A-B-C and Y-A-B-D are both simple sequences when taken separately. But when the system needs to memorize both sequences, A-B becomes a repeating sequence, and as a result both become complex sequences . We define the *degree of a set of sequences* as the maximum length of all the shortest prior sequences that uniquely determine all the components of all the sequences in the set. Because the first component of each sequence is its unique identifier, the definition of the set degree does not depend on how this set of sequences is ordered. Moreover, the degree of a set of sequences must be smaller than the length of the longest sequence in the set. With this definition, it is sufficient to satisfy the condition of Proposition 3 if the number of SR units in each assembly is greater than or equal to the set degree.

The sequential learning procedure is not necessary for the validity of Proposition 4. A more natural procedure of sequential training is postpone retraining until interfered sequences need to be recalled in a specific application. This procedure is more consistent with the process of human learning. One often does not notice memory interference until being tested in a psychological experiment or daily life. This learning procedure blurs the difference between learning a new

sequence for the first time and relearning an interfered sequence. Proposition 4 essentially implies that more and more sequences will be acquired by the system as the learning experience of the model extends. This is an important point. As a result, the anticipation model can be viewed as an open learning system. No rigid procedure for sequential training is needed for the system to increase its long-term memory capacity. The model automatically increases the capacity by just focusing on learning the current sequence. Also, the number of detectors needed to satisfy Proposition 4 can be significantly smaller than the upper limit of $\sum_{i=1}^{k}(|S_i|-1)$ for k sequences. This is because detectors can be shared within the same sequence as well as across different sequences. This will be further discussed in the next section.

4 Simulation Results

To illustrate the model's capability in learning an arbitrarily complex sequence, we show the following computer simulation for learning input sequence <TO - BE - OR - NOT - TO - BE>. The simulated network has 24 detector units, 24 terminals, and 6 SR units for each shift-register assembly (144 SR units in total). Figure 2 shows the activity trace of the network from a simulation run. We use symbol '#' as the end marker, and symbol '-' as a distinct symbol separating meaningful words. The network learned the sequence in 5 training sweeps. In the last training sweep, the system correctly anticipates 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, used to set r.

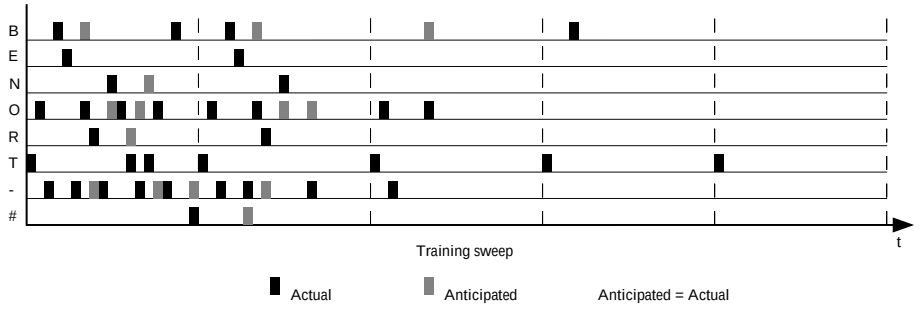

Fig. 2. Training and generation of the sequence <TO-BE-OR-NOT-TO-BE>. The activity traces of the input terminals are shown, where a black box represents an actual terminal activity, a gray box represents an anticipated activity that does not match the actual input, and a white box represents a match between an anticipated and an actual input activity. The training phase takes 5 training sweeps. The generation process is initiated by presenting the first component of the sequence, T. The parameter values used are: $\alpha = 0.2$, $\delta = 0.1$, and $C = 3.0$.

Once one sequence is learned, a right subsequence of the sequence can be generated from a middle point of the sequence. In the above example, with symbol R as the initial input the network will correctly generate 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 generate the remaining part of the sequence. A subsequence can activate an detector which then produces a certain component. The component can then join the subsequence 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 continue a familiar song or a piece of music being exposed to a part of it.

Proposition 4 guarantees that the anticipation model does not suffer from catastrophic interference. Interference exists nonetheless in sequential training, because committed detectors may be seized by later training or retraining to make different anticipation. For example, assume that the system is sequentially trained with two simple sequences, S_a: C-A-T and S_b: E-A-R. After S_a is learned, the training with S_b will lead to the following situation. The previously established link from A to T will be replaced by a link from E-A to R. Thus, S_a is interfered and cannot be generated after S_b is acquired. The critical question is what kind of interference is exhibited by the model, and how severely does it affect learning performance? The extent of interference depends on the amount of overlap between the sequence to be learned and the sequences already stored in the memory. Clearly, if a new sequence has no component in common with the stored sequences, the sequence can be trained as if nothing had been learned by the model. In this sense, interference is caused by the similarity between the sequence and the memory. This is consistent with psychological studies on retroactive interference (see Sect. 1.2).

Knowing that the amount of interference, and thus retraining, depends on the overlap of the sequences to be learned, we wanted to evaluate the system by arbitrarily selecting a domain that contains a lot of overlaps among the sequences. The database of the sequences used consists of the titles of all sessions that were held during the 1994 *IEEE International Conference on Neural Networks* (ICNN-94). This database has 97 sequences, as listed in Table 1. These titles are listed without any change and in exactly the same order as they appear in the final conference program, even retaining the obvious mistakes printed on the program. Evident from the table, there are many overlapping subsequences within the database. Thus, these sequences provide a good testbed for evaluating sequential training and retroactive interference.

For training with this database, the chosen network has 131 input terminals - 34 for the symbol set (26 English letters plus "#", " " (space), ".", "&", "?", "-", ":", and "/") and 97 for the identifiers of the 97 sequences. Each SR assembly contains 40 units. Also, the network needs at least 1,088 detectors and 1,088 modulators. Hence, the network has a total of 7,567 units. The parameters of the network are: $\alpha = 0.2$, $\delta = 1/40$ and $C = 535$. To measure the extent of interference, we record the number of retraining sweeps required to eliminate

Table 1. Sequence Base for Sequential Training

no.	Sequence	no.	Sequence
1	Social & philosophical implications of computational intelligence	50	Image recognition
2	Neurocontrol research: real-world perspectives	51	Medical applications
3	Fuzzy neural systems	52	Parallel architectures
4	Advanced analog neural networks and applications	53	Associative memory I
5	Neural networks for control	54	Pattern recognition IV
6	Neural networks implementations	55	Supervised learning III
7	Hybrid systems I.D.	56	Learning and memory IV
8	Artificial life	57	Intelligent control IV
9	Learning and recognition for intelligent control	58	Economic/Finance/Business applications
10	Artificially intelligent neural networks	59	Machine vision I
11	Hybrid systems II	60	Machine vision
12	Supervised learning X	61	Architecture I
13	Intelligent neural controllers: algorithms and applications	62	Supervised learning V
14	Who makes the rules?	63	Speech I
15	Pulsed neural networks	64	Robotics
16	Fuzzy neural systems II	65	Associative memory II
17	Neural networks applications to estimation and identification	66	Medical applications II
18	Adaptive resonance theory neural networks	67	Modular/Digital implementations
19	Analog neural chips and machines	68	Pattern recognition VI
20	Learning and memory I	69	Robotics II
21	Pattern recognition I	70	Unsupervised learning I
22	Supervised learning I	71	Optimization I
23	Intelligent control I	72	Applications in image recognition
24	Neurobiology	73	Architecture III
25	Cognitive science	74	Optimization II
26	Image processing III	75	Supervised learning VII
27	Neural network implementation II	76	Associative memory IV
28	Applications of neural networks to power systems	77	Robotics III
29	Neural system hardware I	78	Speech III
30	Time series prediction and analysis	79	Unsupervised learning II
31	Probabilistic neural networks and radial basis function networks	80	Neurodynamics I
32	Pattern recognition II	81	Applications I
33	Supervised learning II	82	Applied industrial manufacturing
34	Image processing I	83	Applications II
35	Learning and memory II	84	Architecture IV
36	Hybrid systems III	85	Optimization III
37	Artificially intelligent networks II	86	Applications in image recognition II
38	Fast learning for neural networks	87	Unsupervised learning III
39	Industry application of neural networks	88	Supervised learning VIII
40	Neural systems hardware II	89	Neurodynamics II
41	Image processing II	90	Computational intelligence
42	Nonlinear PCA neural networks	91	Optimization using Hopfield networks
43	Intelligent control III	92	Supervised learning IX
44	Pattern recognition III	93	Applications to communications
45	Supervised learning III	94	Applications III
46	Applications in power	95	Unsupervised learning IV
47	Time series prediction and analysis II	96	Optimization IV
48	Learning and memory III	97	Applications
49	Intelligent robotics		

all interference for every round of sequential training. This number is a good indicator of how much retraining is needed to store all of the sequences that have been sequentially presented to the system. Also, we record the number of intact uninterfered (intact) sequences right after the acquisition of the latest sequence. Figure 3 shows the number of intact sequences and the number of retraining sweeps plotted against training rounds.

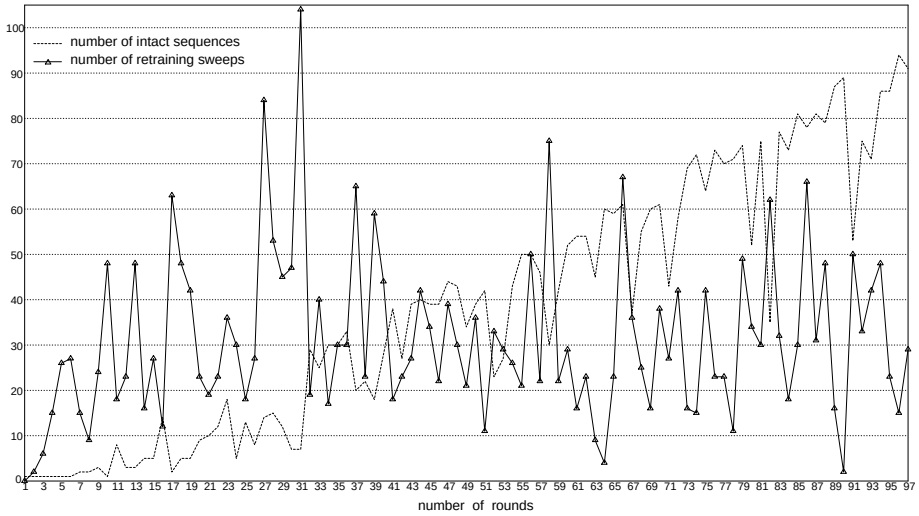

Fig. 3. The number of intact sequences and the number of retraining sweeps with respect to training rounds during training with the Table 1 database.

Several conclusions can be drawn from this simulation result. The first and the most important conclusion is that the number of retraining sweeps seems independent of the size of the previous memory. The overall curve for retraining sweeps remains flat, even though there is large variation across different training rounds. We view this result as particularly significant because it suggests that, in the anticipation model, the amount of interference caused by learning a new sequence does not increase with the number of previously memorized sequences. This conclusion not only conforms intuitively with human performance of long-term learning, but also makes the model feasible to provide a reliable sequential memory that can be incrementally updated later on. Hence, new items can be incorporated into the memory without being limited by those items already in the memory. We refer to this property of sequential learning/memory as *capacity-independent incremental learning/memory.* The anticipation model exhibits this property because a learned sequence leaves its traces across the network, involving a set of distributed associations between subsequences and context detectors (see Fig. 1). On the other hand, each context detector stores its context locally. When a new sequence is learned, it employs a group of context

detectors, some of which may have been committed, thus causing interference. But as the sequential memor y becomes large, so is the number of context detectors. Out of these detectors, only a certain number of them will be interfered as a result of learning a new sequence. The number of interfered detectors tends to relate to the new sequence itself, not the size of the sequential memory . Contrasting capacity-independent incremental learning , catastrophic interference would require simultaneous retraining of the entire memory when a new item is to be learned. The cost of retraining when catastrophic interference occurs can be prohibitive if the size of memory is not so small. Ruiz de Angulo and Torras (1995) presented a study on sequential learning of multilayer perceptrons, and reported that their model can learn sequentially several most recent patterns. Although it is a better result than original multilayer perceptrons, their model appears unable to support a sizable memory. The high variations in the number of retraining sweeps are caused by the overlaps between the stored sequences. For long overlapping subsequences, many sweeps may be needed to resolve the interference caused by overlaps.

The second conclusion is that the number of intact sequences increases with rounds of sequential training approximately linearly. This is to be expected given the result on the amount of retraining. Again, there are considerable variations from one round to another. Since interference is caused by the overlap between a new sequence and the stored sequences, another way of looking at this result is the following. As the memory expands, relatively the fewer items in the memory will overlap with the new sequence.

Figure 4 illustrates the detailed retraining process during round 96. Right after the model has learned S_{96} of Table 1, the only interfered sequence is S_{14}. After S_{14} is retrained, S_{96} is interfered and has to be retrained. This finishes the first retraining cycle for round 96. During the second cycle, S_{71} is found to be interfered. After S_{71} is retrained, S_{96} is interfered again and has to be retrained. The retraining with S_{71} alternates with that of S_{96} for three more cycles. Notice the large overlap between S_{71}: "*OPTIMIZATION* I" and S_{96}: "*OPTIMIZATION* IV". In cycle 6, several more sequences are interfered. After retraining them sequentially, all of the first 96 sequences have been learned successfully.

To examine the amount of detector sharing, we compare the detector use in the anticipation model with one in which no detector is shared by different components (see for example Wang & Arbib, 1993). Without detector sharing, the number of detectors needed to acquire all of the 97 sequences is $\sum_{i=1}^{97}(|S_i|-1)$, which equals 2520. As stated earlier, the anticipation model needs 1,088 detectors to learn all of the sequences. Thus, the detector sharing in the anticipation model cuts required context detectors by nearly a factor of 2.5.

5 Context Learning by Chunking

Even though the amount of retraining does not depend on the number of stored sequences in the memory, retraining can still be expensive. As shown in Fig. 3, it usually takes dozens of retraining sweeps to fully incorporate a new sequence.

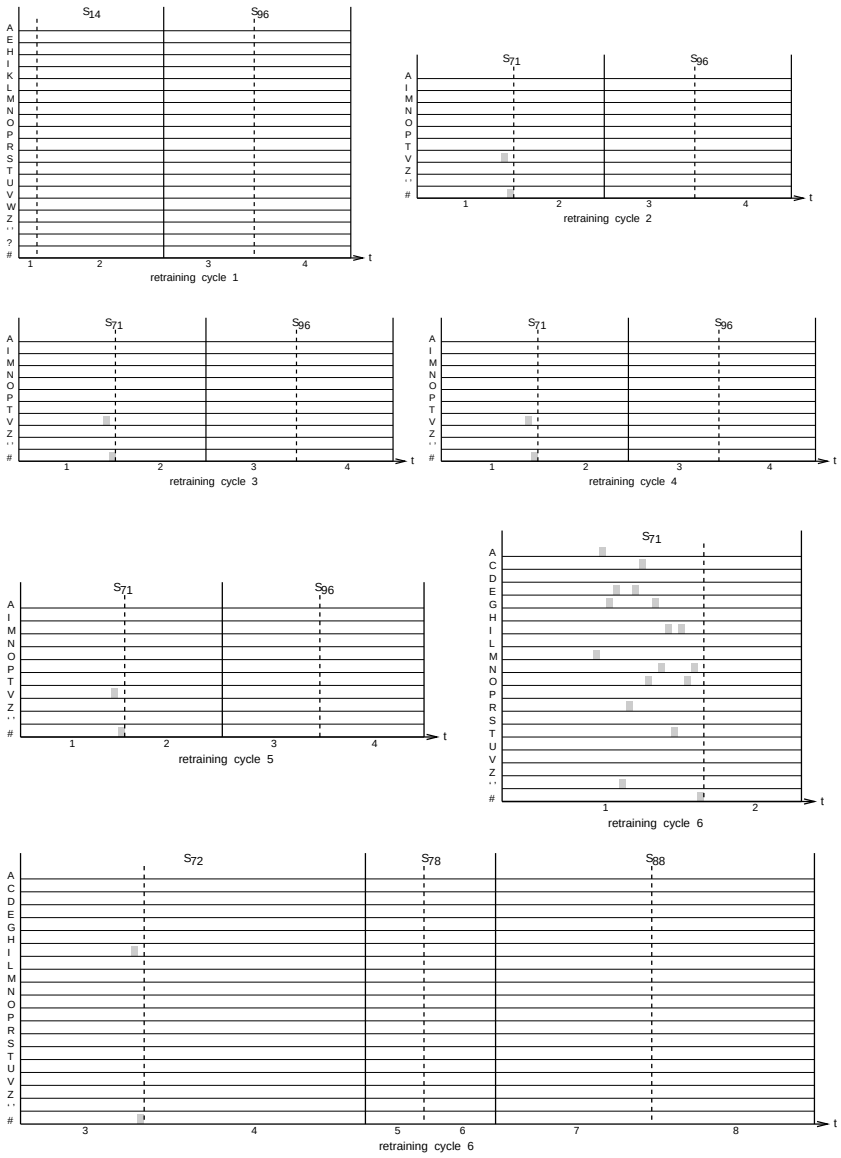

Fig. 4. Retraining process during round 96 of Fig. 3. Each row represents the activity trace of an input terminal indicated by the corresponding input symbol. White boxes represent correctly anticipated terminal activities in sequence generation, whereas gray boxes represent terminal activities which do not match anticipated ones. Solid vertical lines separate training sweeps of different sequences, and dash vertical lines separate training sweeps of the same sequence. Cycle numbers are indicated under each panel. Time runs from left to right.

Our analysis of the model indicates that repeated sweeps are usually caused by the need to commit a new detector and gradually expand the context of that detector in order to resolve ambiguities caused by long recurring subsequences. For example, consider the situation that the system has stored the following two sequences

S_c: "*JOE LIKES* 0"

S_d: "*JAN LIKES* 1",

and the network is to learn the sequence

S_e: "*DEB LIKES* 2".

There is an overlapping subsequence between S_c and S_d: "*LIKES*". A common situation before S_e is learned is that there are two detectors, say u_1 and u_2, tuned to the contexts of "*E LIKES*" and "*N LIKES*", respectively. While S_e is being trained, neither u_1 nor u_2 can be activated and a new detector, say u_3, must be committed to anticipate "2" (u_3 needs to recognize only "*S*" since "*S*" cannot activate either u_1 or u_2). Suppose now the network is to learn yet another sequence,

S_f: "*DIK LIKES* 3".

S_e and S_f will take turns to capture u_3 and gradually increase its degree until u_3 can detect either "*B LIKES*" or "*K LIKES*". Eventually, another detector, say u_4, will be committed for the other sequence. This gradual process of degree increment is the major factor causing numerous retraining sweeps.

The above observation has led to the following extended model for cutting the amount of retraining. The basic idea is to incorporate a chunking mechanism so that newly committed detectors may expand their contexts from chunks formed previously, instead of from the scratch.

The extended model consists of a dual architecture, shown in Figure 5. The dual architecture contains a *generation network* (on the left of Fig. 5), which is almost the same as the original architecture (see Fig. 1), and another similar network, called the chunking network (on the right of Fig. 5). The two networks are mutually connected at the top. The *chucking network* does not produce anticipation, and thus does not need a layer of modulators. Because of this, the detectors in this network do not increase their degrees by a mismatch. Besides, the chunking network mirrors every process occurring in the generation network.

At any time step during training, there is a pair of winning detectors in the dual architecture, each corresponding to one network. The algorithm is designed so that the winning detector of the chunking network has a degree less by 1 than the degree of the winning detector of the generation network. In addition, a newly committed detector of the generation network may take a degree which is 1 plus the degree of the activated chunk detector. We refer to the detectors of the chunking network as *chunk detectors*. The interaction between the two networks takes place via the two-way connections between the two networks

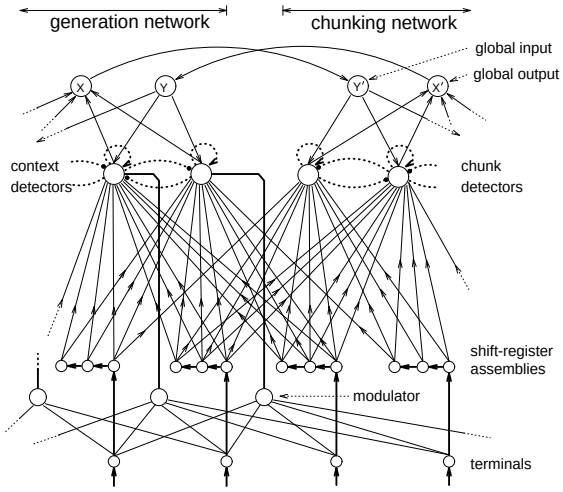

Fig. 5. Dual architecture of the extended anticipation model for chunking. The architecture consists of two mutually connected networks: the generation network and the chunking network. The connections to and from global input and output units have fixed weights. See the caption of Fig. 1 for other notations.

(Fig. 5). The introduction of the chunk detectors can speed up learning when a subsequence (a chunk) occurs multiple times in the input flow. More specifically, assume that a context detector u_i of degree d has learned to recognize a context, and a corresponding chunk detector $u_{i'}$ of degree $d-1$ has learned a chunk which is a right subsequence of the context learned by u_i. If the chunk occurs at least twice, then there will be a time when $u_{i'}$ is activated but u_i is not. Through the learning process an uncommitted context detector, say u_j, is activated (thus committed). Instead of starting from degree 1, u_j starts its degree at the value of d, leading to a significant reduction of training/retraining sweeps. The speedup in sequential training comes with a cost. Obviously, the addition of another network - the chunking network - adds both to the size of the overall network and to additional computing time. The formal description of the chunking network is given in Wang and Yuwono (1996).

Before presenting simulation results with chunking, we explain using the earlier example how the dual architecture helps speed up training. After training with S_c and S_d, there will be a context detector tuned to either "*E LIKES*" or "*N LIKES*". This, in turn, will lead to the formation of the chunk "*LIKES*"; that is, a chunk detector will be tuned to "*LIKES*" since the detector has a degree that is one less than that of the corresponding context detector. After the chunk is formed, Se can be acquired easily - the detector that is trained to associate with "2" can obtain its appropriate degree in just one sweep, thanks to the formation of the chunk "*LIKES*". Similarly, S_f can be acquired quickly.

To show the effectiveness of the chunking network, we present simulations using the same sequences (Table 1) and the same procedure as in the previous

section. To complete the training, the dual architecture requires 1,234 context detectors as compared to 1,088 without chunking, and 436 chunk detectors. Other parts of the network are the same as used in the previous section. Figure 6 gives the result. A comparison between Fig. 6 and in Fig. 3 shows that the former ends up with only a few more intact sequences. This indicates that, in the dual architecture, later training causes almost the same amount of interference. On the other hand, when the chunking is incorporated, the number of retraining sweeps on the whole is cut dramatically. The total number of retraining sweeps during the entire training is 1,104 when the chunking network is included. This is compared to 3,029 without the chunking network, hence a reduction of the overall amount of sequential training almost by three-fold.

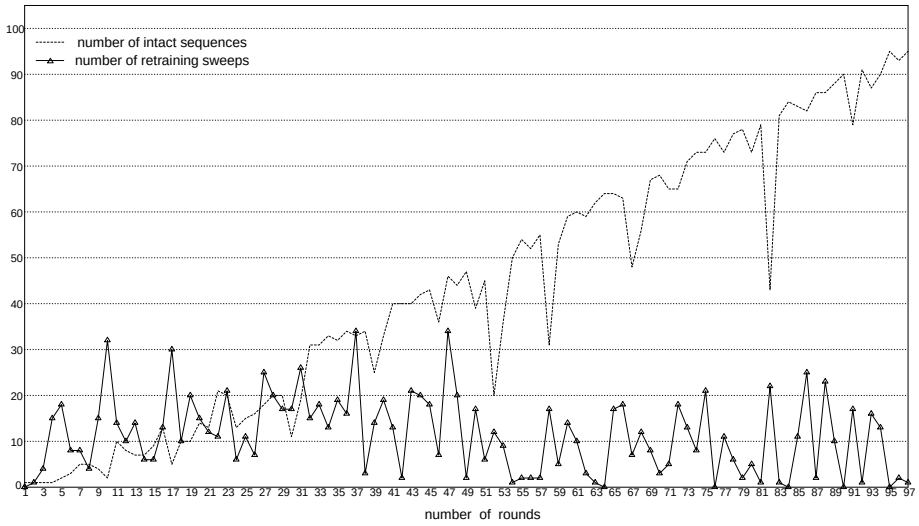

Fig. 6. The number of intact sequences and the number of retraining sweeps with respect to training rounds during training with the Table 1 database with the dual architecture.

For a comparison with Fig. 4 which illustrates the retraining process of round 96, Figure 7 shows the detailed retraining process during round 96 using the dual architecture. Right after S_{96} is learned, three sequences are interfered: S_{59}, S_{60} and S_{86}. After S_{59}: "*MACHINE VISION* I" is retrained, S_{60}: "*MACHINE VISION* II" can be correctly generated without further retraining. This interesting situation arises because the two sequences have a large overlap and it is the overlapping part that is interfered during training S_{96}. Thus, when the overlapping part is regained during retraining with S_{59}, both S_{59} and S_{60} can be recalled correctly. The system needs another sweep to regain S_{86}. After the first retraining cycle, all of the first 96 sequences have been acquired.

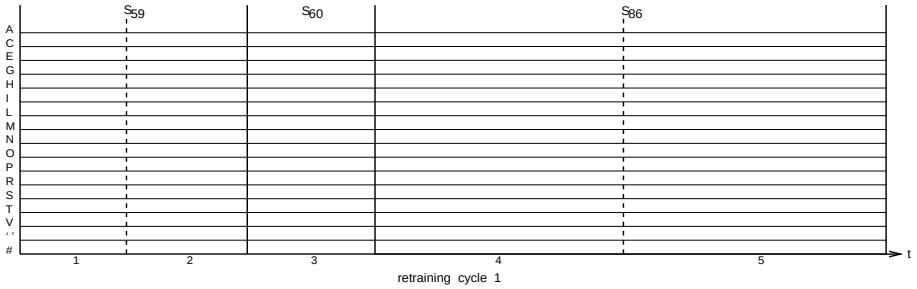

Fig. 7. Retraining process during round 96 of Fig. 6. The interference with S_{60} is eliminated as a result of retraining S_{59}. See the caption of Fig. 4 for notations.

6 Further Discussion

The idea of anticipation-based learning seems to be consistent with psychological evidence on human learning of sequential behaviors. Meyer (1956) noted that expectation is key to music cognition. When a temporal sequence is repeatedly presented to subjects, according to Nissen and Bullemer (1987), the reaction to a particular component in 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 finding rules out the possibility that the reduction in reaction time is due to the familiarity with a component. These findings have been confirmed by later experiments (Willingham et al., 1989; Cohen et al., 1990). The results suggest 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 (Nissen & Bullemer, 1987; Willingham et al., 1989; Curran & Keele, 1993).

As analyzed earlier, the anticipation model does not suffer from catastrophic interference. When multiple sequences are presented to the model sequentially, some degree of interference occurs. But this kind of interference can be overcome by retraining the interfered sequences. Extensive computer simulations indicate that the amount of retraining does not increase as the number of sequences stored in the model increases. The anticipation model is characteristic of capacity-independent incremental learning during sequential training. These results, plus the fact that interference is caused by the overlap between a new sequence and stored sequences, suggest that the behavior of the model in sequential learning resembles aspects of retroactive interference.

After a sequence S is learned, it can be generated by its beginning component - the sequence identifier. Partial sequence generation can also be elicited by a subsequence of S. If a sufficient subsequence is presented, the rest of S can be generated entirely. Partial generation may stop before the rest of the sequence is completed. For example, after the sequence: X-A-B-C-D-E-A-B-C-D-F is

learned, the presentation of A activates the subsequence B-C-D, but not the rest. The anticipation model exhibits partial generation because a sequence is stored as a chain of associations, each of which is triggered by a context, or a subsequence. This property of the model is consistent with our experience that we are able to pick up a familiar song, a melody, or an action sequence (like *Tai Chi*) from the middle.

As shown in Sect. 5, the model's capability of chunking repeated subsequences within a sequence and between sequences substantially reduces the amount of retraining and improves the overall efficiency of learning. Without chunking, recurring subsequences must be learned from the scratch. The basic idea behind current chunking is to learn a recurring subsequence just once and store it as a chunk, so that the next time the subsequence occurs the model can simply use the chunk as a basic component. Chunking is a fundamental characteristic of human information processing (Miller, 1956; Simon, 1974). We note that, though the present model has addressed some aspects of chunking, the general issue of automatic chunking is very challenging and remains an open problem. What constitutes a chunk? A chunk is often taken to be a meaningful subsequence (such as a word), but it may also be just a convenient way of breaking a long sequence into shorter subsequences for facilitating further processing. In the anticipation model, a chunk corresponds to a repeated subsequence. This is a reasonable definition in the present context. The model, through its mechanism of context learning , provides a neural network basis for forming such chunks. On the other hand, this definition of a chunk does not capture the richness of general chunking. Chunking depends critically on the STM capacity (Miller, 1956). Furthermore, different people, may have different ways of chunking the same sequence in order to overcome STM limitations and memorize the sequence. Chunking also depends on general knowledge and custom. For example, we tend to chunk a 10-digit telephone number in the U.S. into three chunks: the first three digits corresponding to an area code, then the next three digits to a district code, and the last four digits. However, the same 10-digit number may well be chunked differently in another country.

Proposition 4 and the property of capacity-independent incremental learning together enable the anticipation model to perform long-term automatic learning of temporal sequences. The system is both adaptive and stable, and its long-term memory capacity increases gradually as learning proceeds. Thus, the anticipation model provides a *sequential memory* , which can store and recall a large number of complex sequences .

7 Summary

In this chapter, we have presented the anticipation model - a neural network model - that learns and generate complex temporal sequences. In the anticipation model, sequences are acquired by one-shot learning that obeys a normalized Hebbian learning rule, in combination with a competitive mechanism realized by a winner-take-all network. During learning and generation, the network ac-

tively anticipates the next component on the basis of a previously anticipated context. A mismatch between the anticipation and the actual input triggers self-organization of context expansion. Analytical results on the anticipation model, presented in Sect. 3, ensure that the model can learn to generate any complex sequences . Multiple sequences are acquired by the model in an incremental fashion, and large-scale simulation results strongly suggest that the model exhibits capacity-independent incremental learning . As a result, the anticipation model provides an effective sequential memory . In addition, by incorporating a form of chunking we have demonstrated significant performance improvement in learning many sequences that have significant overlaps.

Finally, the anticipation model argues (see also Wang & Arbib, 1993) from a computational perspective for the chaining theory of temporal behavior, which was rejected by Lashley (1951) but supported by recent psychological studies of serial order organization (Murdock, 1987; Lewandowsky & Murdock Jr., 1989). Simple associative chaining between adjacent sequence components is too limited to be true. However, if chaining between remote components and chunking of subsequences into high-order components are allowed, the basic idea of associative chaining can give rise to much more complex temporal behaviors, going much beyond what was realized by Lashley (1951). The anticipation model shows how learning and generation of complex temporal sequences can be achieved by self-organizing in a neural network.

References

Baram, Y. (1994). Memorizing binary vector sequences by a sparsely encoded network. *IEEE Transactions on Neural Networks*, **5**(6), 974-981.

Barnes, J. M., & Underwood, B. J. (1959). 'Fate' of first-list associations in transfer theory. *Journal of Experimental Psychology*, **58**, 97-105.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5**(2), 157-166.

Bower, G. H., Thompson-Schill, S., & Tulving, E. (1994). Reducing retroactive interference: An interference analysis. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **20**, 51-66.

Bradski, G., Carpenter, G. A., & Grossberg, S. (1994). STORE working memory networks for storage and recall of arbitrary temporal sequences. *Biological Cybernetics*, **71**, 469-480.

Buhmann, J., & Schulten, K. (1987). Noise-driven temporal association in neural networks. *Europhysics Letters*, **4**, 1205-1209.

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphs, and Imaging Processing*, **37**, 54-115.

Chandler, C. C. (1993). Accessing related events increases retroactive interference in a matching test. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **19**, 967-974.

Cohen, A., Ivry, R. I., & Keele, S. W. (1990). Attention and structure in sequence learning. *Journal of Experimenal Psychology*, **16**, 17-30.

Cohen, M. A., & Grossberg, S. (1987). Masking fields: A massively parallel neural architecture for learning, recognizing, and predicting multiple groupings of patterned data. *Applied Optics*, **26**, 1866-1891.

Crooks, R. L., & Stein, J. (1991). *Psychology: Science, behavior, and life.* Fort Worth, TX: Holt, Rinehart and Winston.

Curran, T., & Keele, S. W. (1993). Attentional and nonattentional forms of sequence learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **19**, 189-202.

Diederich, S., & Opper, M. (1987). Learning of correlated patterns in spin-like glass networks by local learning rules. *Physical Review Letters*, **58**, 949-952.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, **14**, 179-211.

French, R. M. (1994). Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, (pp. 335-340). Hillsdale, NJ: Erlbaum.

Granger, R., Whitson, J., Larson, J., & Lynch, G. (1994). Non-Hebbian properties of long-term potentiation enable high-capacity encoding of temporal sequences. *Proceedings of the National Academy of Sciences of USA*, **91**, 10104-10108.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, **23**, 121-134.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, **11**, 23-63.

Guyon, I., Personnaz, L., Nadal, J. P., & Dreyfus, G. (1988). Storage and retrieval of complex sequences in neural networks. *Physics Review A*, **38**, 6365-6372.

Hebb, D. O. (1949). *The Organization of behavior.* New York: Wiley & Sons.

Hertz, H., Krogh, A., & Palmer, R. G. (1991). *Introduction to the theory of neural computation.* Redwood City, CA: Addison-Wesley.

Heskes, T. M., & Gielen, S. (1992). Retrieval of pattern sequences at variable speeds in a neural network with delays. *Neural Networks*, bf 5, 145-152.

Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, (pp. 531-546). Hillsdale, NJ: Erlbaum.

Kanerva, P. (1988). *Sparse distributed memory.* Cambridge, MA: MIT Press.

Kantor, I., & Sompolinsky, H. (1987). Associative recall of memory without errors. *Physics Review A*, bf 35, 380-392.

Kruschke, J. K. (1992). ALCOVE: An exemplar-based model of category learning. *Psychological Review*, **99**, 22-44.

Lashley, K. S. (1951). The problem of serial order in behavior. In L. A. Jeffress (Ed.), *Cerebral mechanisms in behavior* (pp. 112-146). New York: Wiley & Sons.

Lewandowsky, S., & Murdock Jr., B. B. (1989). Memory for serial order. *Psychological Review*, **96**, 25-57.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, **24**, 109-165.

Meyer, L. B. (1956). *Emotion and meaning in music.* Chicago, IL: University of Chicago Press.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, **63**, 81-97.

Mozer, M. C. (1993). Neural net architectures for temporal sequence processing. In A. Weigend & N. Gershenfeld (Ed.), *Predicting the future and understanding the past* (pp. 243-264). Redwood City, CA: Addison-Wesley.

Murdock, B. B. J. (1987). Serial-order effects in a distributed-memory model. In D. S. Gorfein & R. R. Hoffman (Ed.), *Memory and learning: The Ebbinghaus centennial conference* (pp. 227-310). Hillsdale, NJ: Erlbaum.

Nissen, M. J., & Bullemer, P. (1987). Attentional requirements of learning: Evidence from performance measures. *Cognitive Psychology*, **19**, 1-32.

Ratcliff, R. (1990). Connectionist models of recognition memory: Constraints imposed by learning and forgetting function. *Psychological Review*, **97**, 285-308.

Rinkus, G. J. (1995). TEMECOR: an associative, spatio-temporal pattern memory for complex state sequences. In *Proceedings of World Congress on Neural Networks*, (pp. I.442-I.448). Washington DC:

Rodriguez, W. A., Borbely, L. S., & Garcia, R. S. (1993). Attenuation by contextual cues of retroactive interference of a conditional discrimination in rats. *Animal Learning & Behavior*, **21**, 101-105.

Ruiz de Angulo, V., & Torras, C. (1995). On-line learning with minimal degradation in feedforward networks. *IEEE Transactions on Neural Networks*, **6**, 657-668.

Simon, H. A. (1974). How big is a chunk? *Science*, **183**, 482-488.

Sloman, S. A., & Rumelhart, D. E. (1992). Reducing interference in distributed memories through episodic gating. In A. F. Healy, S. M. Kosslyn, & R. M. Shiffrin (Ed.), *From learning theory to connectionist theory: Essays in honor of William K. Estes* (pp. 227-248). Hillsdale, NJ: Erlbaum.

Sompolinsky, H., & Kanter, I. (1986). Temporal association in asymmetric neural networks. *Physics Review Letters*, **57**, 2861-2864.

von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, **14**, 85-100.

Wang, D. L. (2001). Temporal pattern processing. In M. A. Arbib (Ed.), *Handbook of brain theory and neural networks*, Second Edition (to appear). Cambridge MA: MIT Press.

Wang, D. L., & Arbib, M. A. (1990). Complex temporal sequence learning based on short-term memory. *Proceedings of the IEEE*, **78**, 1536-1543.

Wang, D. L., & Arbib, M. A. (1993). Timing and chunking in processing temporal order. *IEEE Transactions on Systems, Man, and Cybernetics*, **23**, 993-1009.

Wang, D. L., & Yuwono, B. (1995). Anticipation-based temporal pattern generation. *IEEE Transactions on Systems, Man, and Cybernetics*, **25**, 615-628.

Wang, D. L., & Yuwono, B. (1996). Incremental learning of complex temporal patterns. *IEEE Transactions on Neural Networks*, **7**, 1465-1481.

Wang, L. (1999). Multi-associative neural networks and their applications to learning and retrieving complex spatio-temporal sequences. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, **29**, 73-82.

Willingham, D. B., Nissen, M. J., & Bullemer, P. (1989). On the development of procedural knowledge. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **15**, 1047-1060.

Willshaw, D. J., Buneman, O. P., & Longuet-Higgins, H. C. (1969). Nonholographic associative memory. *Nature*, **222**, 960-962.