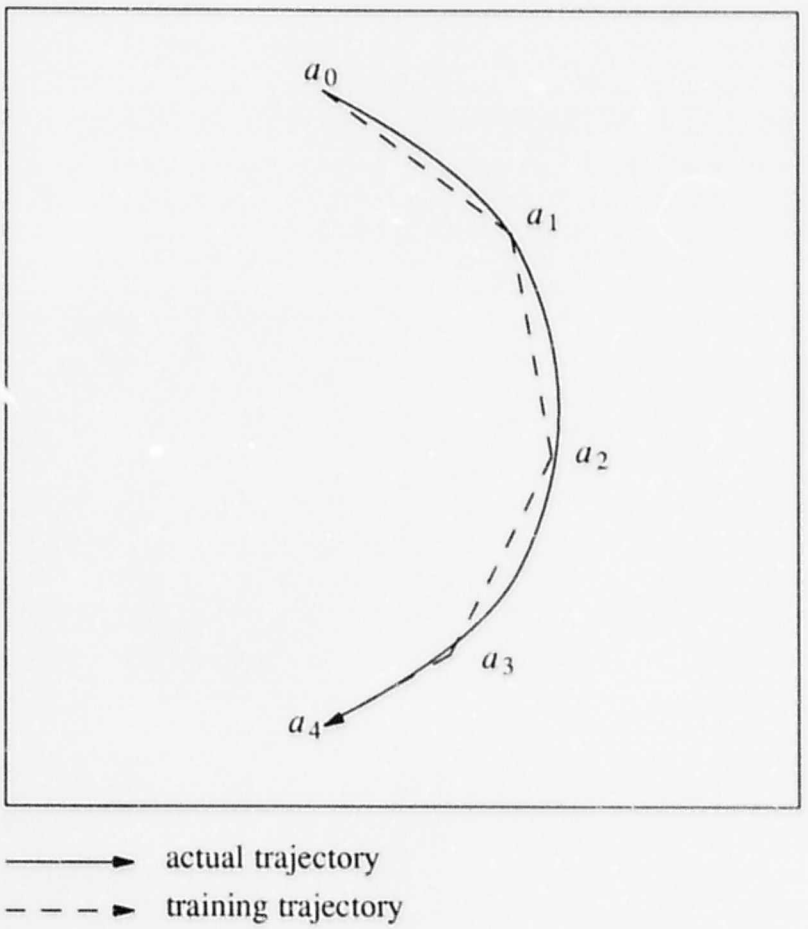



Fig. 5.19 Actual and training trajectories used in a simulation

along the same trajectory but at twice the speed generates a sequence of images with $b_0, b_1, b_2, \ldots$ etc. as the sampling points. Since each of these sampling points occupies a different location in space, objects moving at different speeds give rise to sequences of images with considerable spatial differences between them, even though they move along the same trajectory. As a result, it is possible to tune several temporal modules to classify the objects' movements along the same trajectory according to speed.

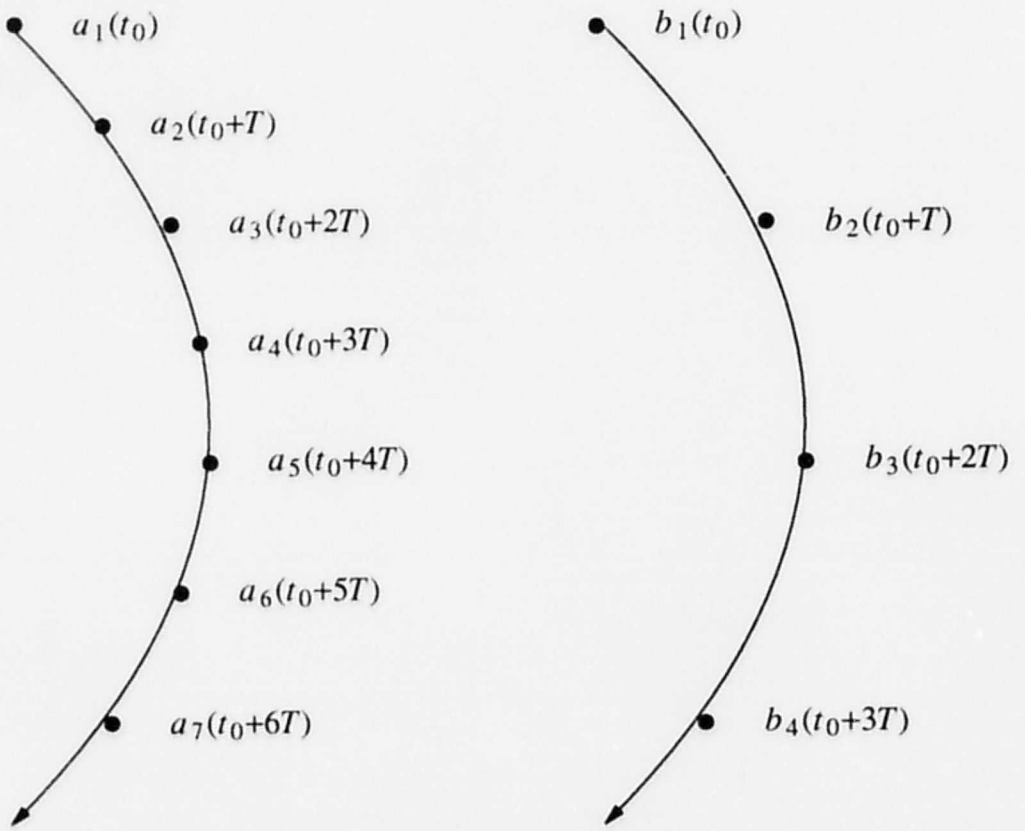



Fig. 5.20 An object moving along the same trajectory but at different speeds.

## 5.8.2. Slightly Altered Trajectories

The performance of a motion detector trained to recognize the movement of an object along different trajectories is evaluated in this section. A motion detector with four temporal modules has been simulated and tested, with each module trained to recognize the movement of an object along a certain trajectory at a speed of 8 pixels per unit time. We shall refer to these trajectories as trajectories 1, 2, 3, and 4, as shown in Fig. 5.21. The size of the moving object is 8×8 pixels.

Fig. 5.22 gives the response of the motion detector with all four modules interconnected when a random-dot pattern with the size of 8×8 pixels moves along trajectory 1. Output 1 shows the gradual rise in the output of the module trained to recognize that trajectory. The module begins

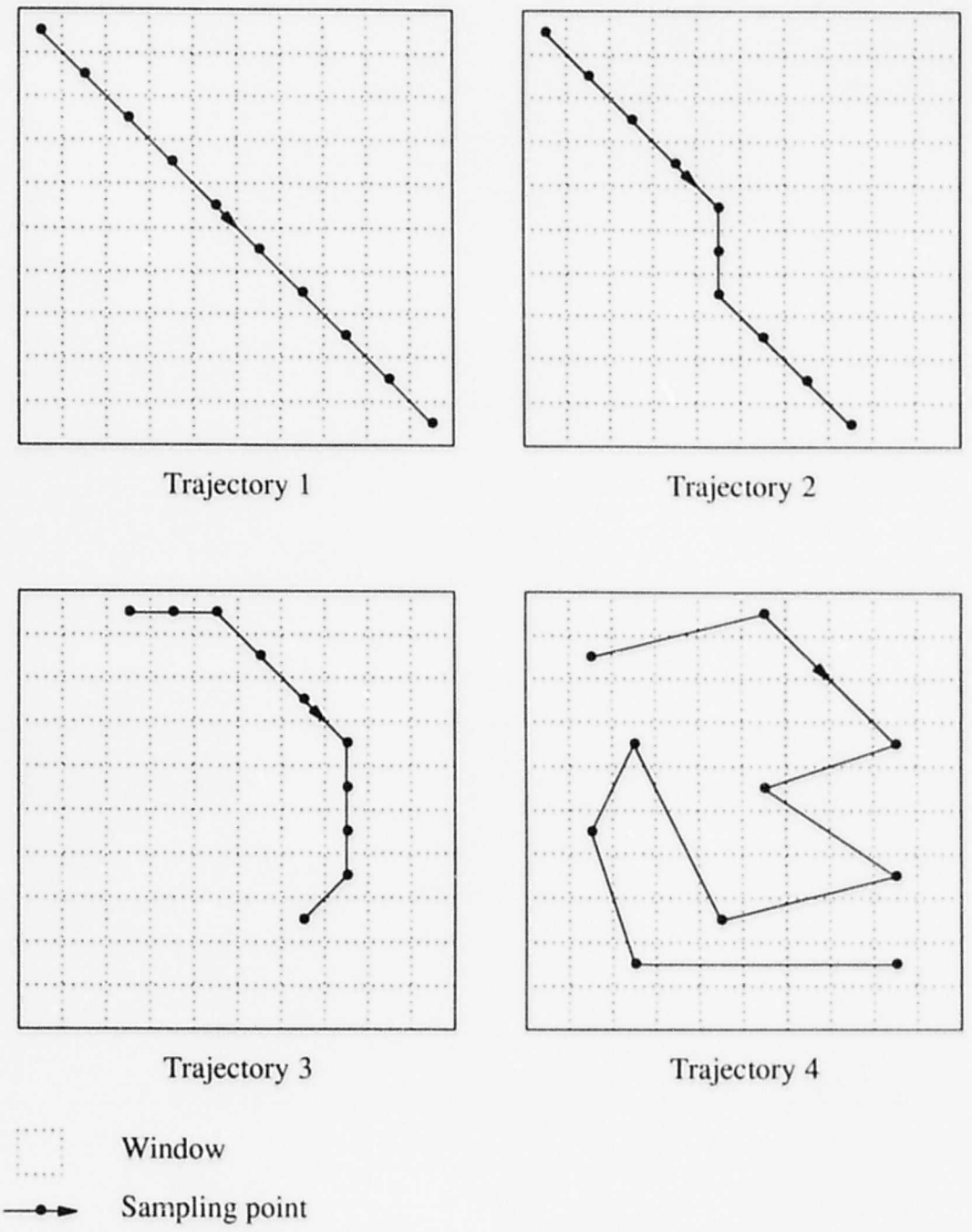



Fig. 5.21 Four training trajectories.

to be activated at about the 3*rd* sample and its output rises above 0.8 after the 6*th* sample, which indicates that the module has recognized the trajectory. Outputs 2, 3 and 4 show the behavior of

the other 3 modules trained to recognize trajectories 2, 3, and 4, respectively. The outputs of modules 3 and 4 stay below 0.1 most of the time because trajectories 3 and 4 are different from trajectory 1. However, the output of module 2 increases gradually until the *6th* sample, and then decreases slowly the rest of the time. The reason for this behavior is that the first few images of the motion sequence for trajectory 2 are quite similar to those images for trajectory 1. The output of module 2 would have continued to increase, but it begins to decrease after the *6th* sample due to the inhibition from module 1.

The competitive interaction between the modules to become a winner has a strong influence on the final output of a module. In order to reach a better understanding of the sensitivity of the detector to position error, we will examine the response of one module with no inhibitory inputs from other modules.

To determine the tolerance of a module to recognize the movement of an object along a different trajectory from the training trajectory, a slightly altered trajectory is used, as shown in Fig. 5.23. As before, the corners of a test trajectory represent the sampling points. For each corner on the training trajectory, four points are defined, each having a positional error $Z$. A test trajectory is generated by randomly selecting one of the four sampling points for each corner of the trajectory.

A training trajectory with $N$ training points would result in the generation of $4^N$ test trajectories. It would be very expensive in computation to perform the tolerance test on all possible trajectories. Instead, a limited test was performed, in which the average output of a module for 10 random samples of the test trajectories was used. The solid and dashed lines in Fig. 5.24 depict the average outputs of two temporal modules at the end of 10 time samples for different values of the positional error $Z$. Two other modules were tested and gave similar results. The simulation results show that the detector can recognize trajectories with positional error of up to 7 pixels, without too much deterioration in performance. The module fails to detect the altered trajectory once the position error $Z$ is greater than 9 pixels.

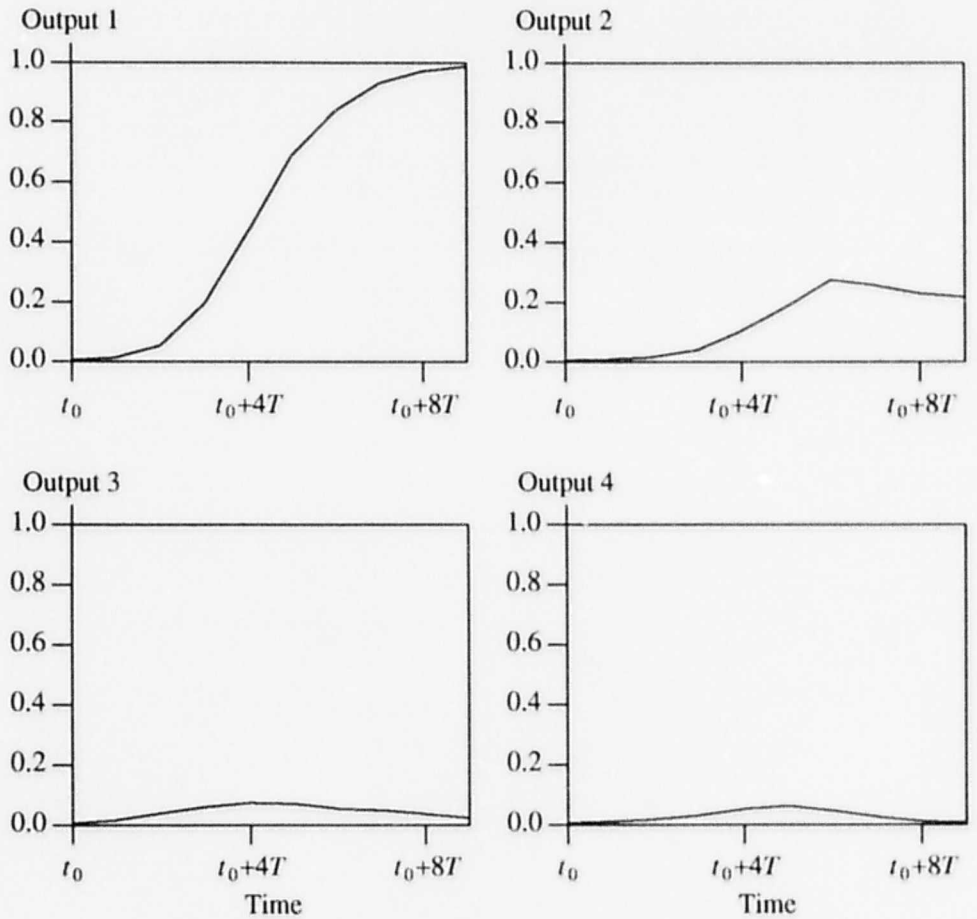



Fig. 5.22 Response of motion detector to a moving object.

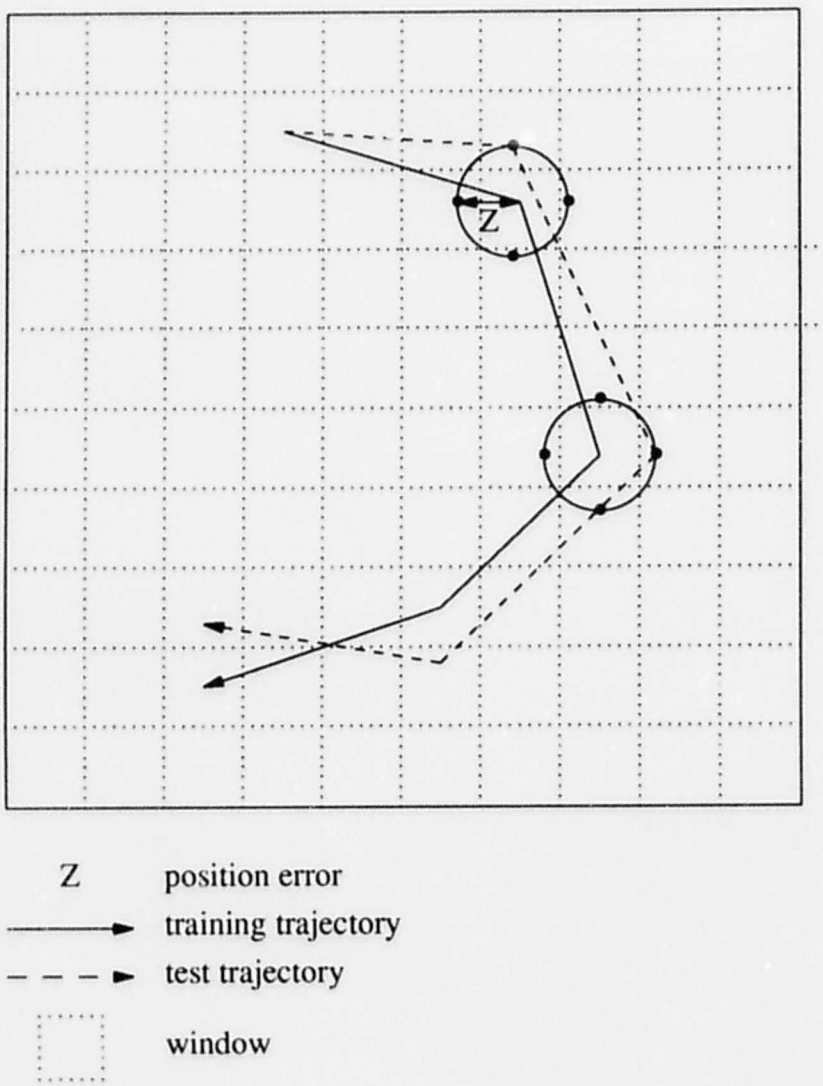



Fig. 5.23 Training and test trajectories used in a simulation

Therefore, the above results clearly show that a trained network can still recognize the motion of an object moving along a slightly altered trajectory. It is important to note that the result presented in the Fig. 5.24 will still be true when the modules are operated in a winner-take-all fashion, as long as the trajectories are far apart and do not interfere with each other. Otherwise the result is no longer true.

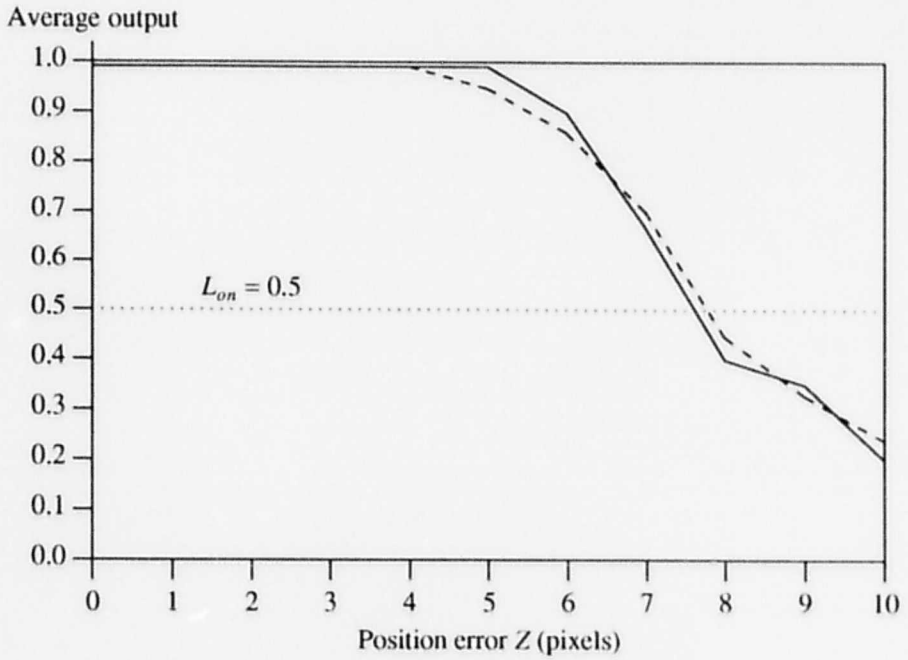



Fig. 5.24 Average outputs of two temporal modules shown as solid and dashed lines versus position error $Z$ from its training trajectory.

## 5.8.3. Objects of Different Physical Sizes

A motion detector is trained to recognize the movement of an object of certain physical size along a particular trajectory. Thus, it is important to know the sensitivity of the motion detector to the movement of objects with physical sizes different from the training size.

The motion detector from the previous section was trained to recognize the motion of a 8×8 pixels object along 4 different trajectories shown in Fig. 5.21. Now, to study the effect of object size on the performance of the detector, objects of different sizes moving along these trajectories are presented.

Fig. 5.25 presents the average outputs of two of the temporal modules as a function of object size. The simulation results show that a module can detect the movement of any object with a size between 3×3 and 14×14 pixels. The other two modules were tested and gave a similar output

response to object size. Comparison of Fig. 5.25 with Fig. 5.9 shows that simulation results derived from a larger network fit the result from the performance study presented earlier. These figures indicate that the detector is not sensitive to the motion of an object whose size is slightly different from the training size due to the blurring action of the summers.

## 5.8.4. Partial Sequences

The simulation results presented so far assumed that a complete sequence is always presented to the motion detector. However, sequences encountered by a detector are quite often partial sequences. A partial sequence is regarded as a fraction of a complete correct sequence containing all the images belonging to that part of the sequence. A partial sequence may be embedded into random sequences, and it may start at any time. Therefore, the system should remain in the reset

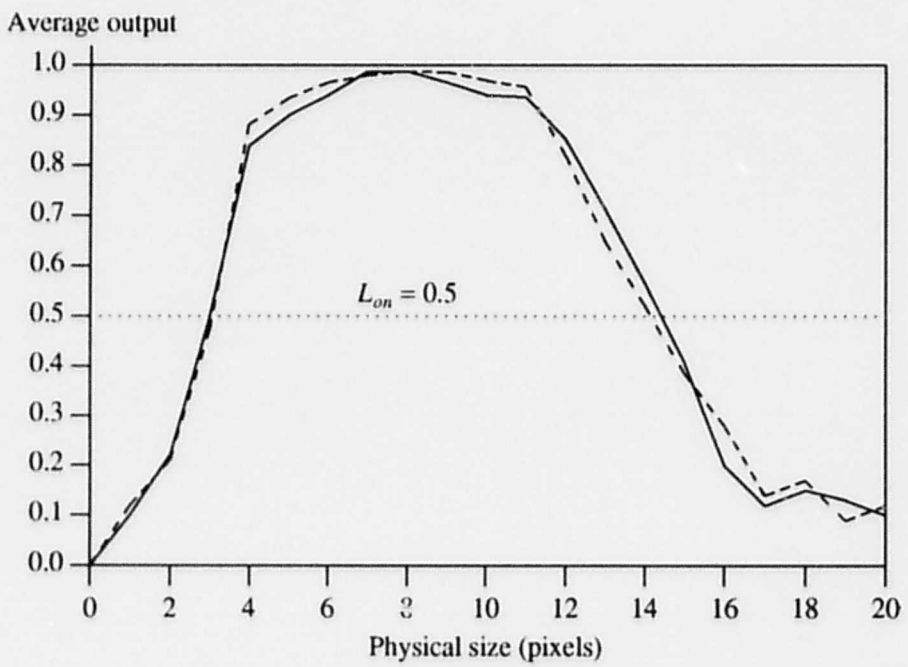



Fig. 5.25 Average outputs of two temporal modules shown as solid and dashed lines versus the size of an object

state during a random input sequence. As soon as the beginning of a partial sequence appears at the input, the system should exit the reset state and start the recognition process. The detector should be able to recognize sequences and generate an output to indicate that only a part of the sequence has been encountered.

To check the ability of a temporal module to recognize a partial sequence, a sequence is used that consists of a random sequence, followed by a correct partial sequence of varying length and finally a random sequence. The test is performed under the condition that the object size is kept the same as the training size and the object moves along the training trajectory. A partial sequence can belong to any part of a correct sequence. Hence, a module was tested with partial sequences of different lengths taken from a correct sequence at random. At the end of the five tests, an average output was obtained. Fig. 5.26 shows the average outputs of two temporal modules when partial sequences of different lengths are presented. The simulation results indicate that the detector is able to recognize partial sequences comprising more than 50% of the images in a complete sequence. The detector fails completely when less than 30% of a complete sequence is presented.

The results in Fig. 5.26 can be explained as follows. Each temporal module has been trained to examine a minimum of 40 % of a correct sequence before an output of 1 is generated at the result node. As a result, when less than 30% of a complete sequence is presented, the result node will not be activated.

### 5.8.5. Input Noise

Noise is an inevitable phenomenon in the operation of any system, and it comes in many forms [Widrow 1985]. Here, we shall consider two common forms of noise that may be encountered during the operation of a motion detector. First, noise can be induced by error in sampling the input, causing erroneous input data. Second, noise in the image input can be caused by random changes in some of the pixels representing the object or the background image. Since the presence of noise in the input cannot be avoided, it is essential to determine the tolerance of a motion detector to noisy input. The detector should be stable in a noisy environment, and should degrade

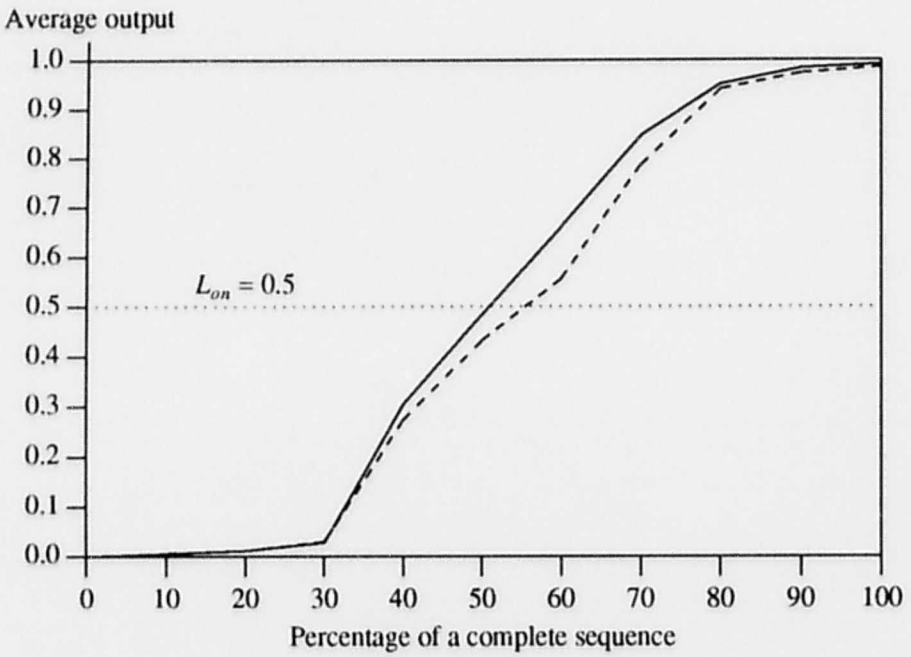



Fig. 5.26 Average outputs of two temporal modules shown as solid and dashed lines versus the percentage length of a partial sequence.

gracefully with increasing noise.

We shall begin by examining the degradation effect of the intermittent loss of correct image input in a sequence due to sampling error. The test sequence is derived by taking a correct sequence and replacing some of the images in the sequence by some random images. The length of the test sequence is kept the same as the length of a correct sequence. For each percentage loss, a module is tested with five test sequences and an average output is obtained. The average outputs of two temporal modules to sequences with different percentages of intermittent loss of correct image input are shown in Fig. 5.27. The simulation results show that the module is able to function correctly when not more than 40 % of the image input is lost. The performance of the detector degrades as the intermittent loss of images increases beyond 40 %.

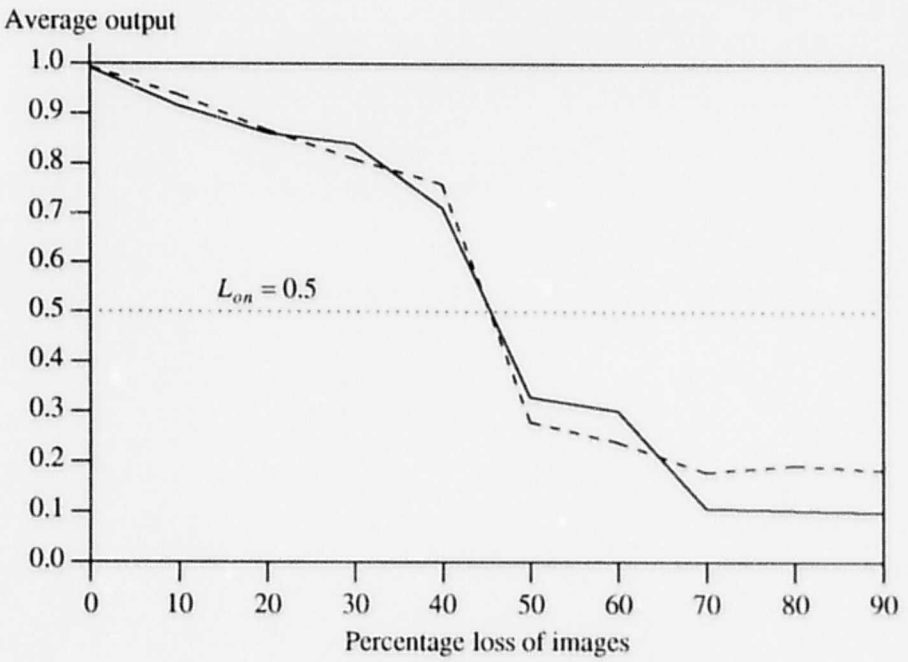



Fig. 5.27 Average outputs of two temporal modules shown as solid and dashed lines versus percentage loss of images in a sequence.

Next, we shall examine the effect of random background noise on the performance of a temporal module to recognize a motion sequence. To perform the test, noise is injected into the image input by randomly inverting the state of a number of pixels in the back plane while a temporal module is being tested to detect the movement of an object. The trajectory, speed and physical size of the object are kept the same as those used during training. Fig. 5.28 shows the average output of two temporal modules versus the percentage of random noise in the back plane. The simulation results indicate that the module can tolerate up to 20 % of background noise.

## 5.9. Discussion

The performance study of a small motion detector indicates that the detector is able to recognize the movement of an object whose size and speed are similar to the training size and the

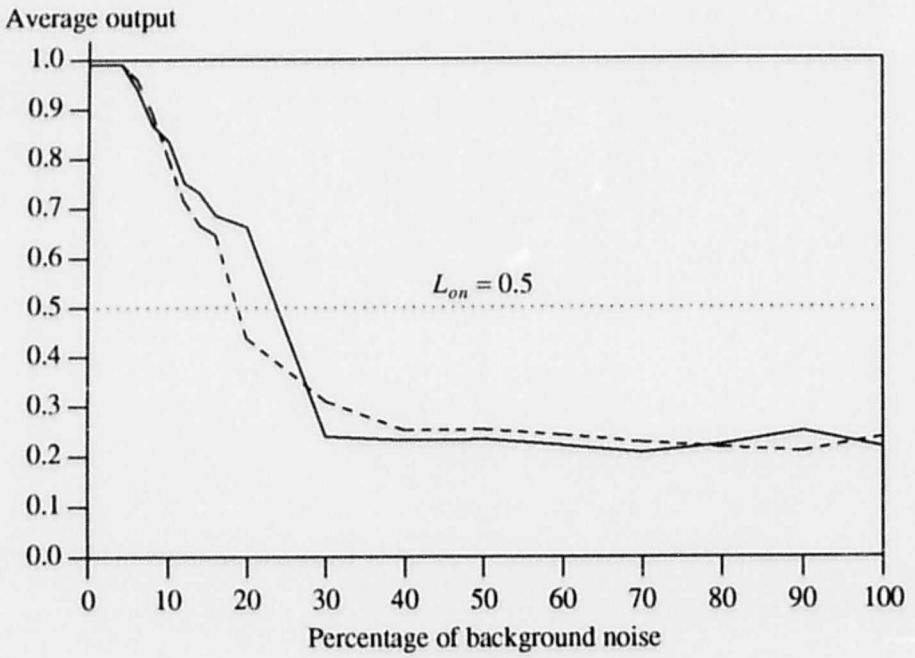



Fig. 5.28 Average outputs of two temporal modules shown as solid and dashed lines versus percentage of background noise level.

training speed, respectively. Even though the study was performed by moving a solid object against a white background, the dynamic behavior under this condition is similar to that of the motion of a random-dot object against a random-dot background in most cases. Different behavior arises when the size of the object is larger than than the window size. Therefore, the results obtained from the performance study can be used to interpret the simulation results obtained from the testing of larger networks when the motion of a random-dot object and background are used as input stimulus.

The simulation results of larger networks show that the proposed motion detector can be trained to recognize a variety of temporal sequences resulting from the motion of an object along different trajectories and at different speeds. A trained detector can detect the movement of an object as long as the speed of motion is close to the training speed, the object has a similar physical

size to that of the training pattern, and the object moves along a trajectory that is only slightly altered from the training trajectory. The detector is able to detect partial motion sequences and is tolerant to input noise.

## 5.10. Conclusion

This chapter has presented an architecture of a motion detector based on several temporal modules presented in chapter 3. The application in motion detection has provided realistic inputs for testing the proposed module. Simulation results show that a module based on a sequential network can be used to keep track of each image in a temporal sequence, and temporal integration of the result from recognition of each image in a sequence can be used to generate an output representing the sequence. The results also show that the module is capable of recognizing partial sequences and is tolerant to noise and other forms of degradation in the input sequence.

# Chapter 6

# Architectural Enhancement

## 6.1. Introduction

A modular neural network was proposed in chapter 3. A temporal module can be enlarged to recognize a long sequence by putting more sequence nodes into a module. However, the limited number of transistors and interconnections on a chip limits a temporal module to a few nodes [Mead 1980, Mead 1989]. As a result, each module has a finite temporal window of size $N$, where $N$ is the number of sequence nodes. Such a module is not suited for recognition of a long sequence with more than $N$ events.

This chapter presents ways to expand the temporal pattern recognition capabilities of the network beyond that of a single module. A modified module is introduced, which can be used to construct a larger network in one of two ways. A number of modules may be concatenated to form a chain capable of recognizing a long sequence of events. Alternatively, modules may be organized in a hierarchical structure to recognize a sequence of sequences of events. For example, if each module is trained to recognize a word, the network may be trained to recognize a sentence.

The chapter begins by presenting concatenated connections, followed by hierarchical connections. Then, simulation results are given to illustrate the networks' capabilities.

## 6.2. Concatenation of Temporal Modules

One approach to recognition of a long sequence is to cascade a number of smaller temporal modules in series, so that each module is only responsible for recognition of a portion of the sequence. Result from recognition of each portion by the respective module is temporally

integrated to generate an output representing the overall sequence. To facilitate the concatenation of several temporal modules, two external connections, which were not present in the basic temporal module described in chapter 3, are provided, as shown in Fig. 6.1. They enable several modules to be concatenated by connecting the input of the first sequence node, $S_1$, to the output of the last sequence node, $S_N$, of the previous module, via an excitatory connection. A linear sum node (*LS*) and its output (*Sum*) have also been added, the need for which will become apparent shortly.

Concatenation of several temporal modules as shown in Fig. 6.2 forms a long chain of sequence nodes, each of which trained to recognize only one event. When a correct sequence is presented to the network, the first event causes node $S_1$ of module $M_1$ to be partially activated. Subsequent events cause downstream nodes to be activated to higher levels, because of the excitatory connections in the forward direction. When a sufficient number of sequence nodes within a module are activated, the result node of that module becomes active, indicating that the subsequence for which this module is responsible has been recognized. Thus, it would be possible to determine that the entire sequence has been received correctly by temporally integrating the outputs of the result nodes in all the modules (this is not shown in the figure).

Consider now the case of a partial sequence that overlaps two cascaded modules, as shown in Fig. 6.3. Modules $M_j$ and $M_{j+1}$ are trained to recognize sequences $l_{s1} = \{E_1, E_2, E_3, E_4\}$ and $l_{s2} = \{E_5, E_6, E_7, E_8\}$, respectively. The test sequence presented is $t_s = \{ E_3, E_4, E_5, E_6\}$. The part of the sequence seen by each module may not have a sufficient number of events to activate its internal result node, which has a non-linear response. Hence, direct temporal integration of the outputs of the two result nodes is not sufficient to recognize this partial sequence. The linear sum node, *LS*, has been introduced to solve this problem. Unlike the result node, the linear sum node has a linear output response representing the number of correct events seen by a module. Thus, in the above case, direct temporal integration of the outputs of the two *LS* nodes will generate an output representing the total length of the partial sequence observed by the two modules. This integration is performed by another node called the *External Result* node (*ER*) in Fig. 6.2. The

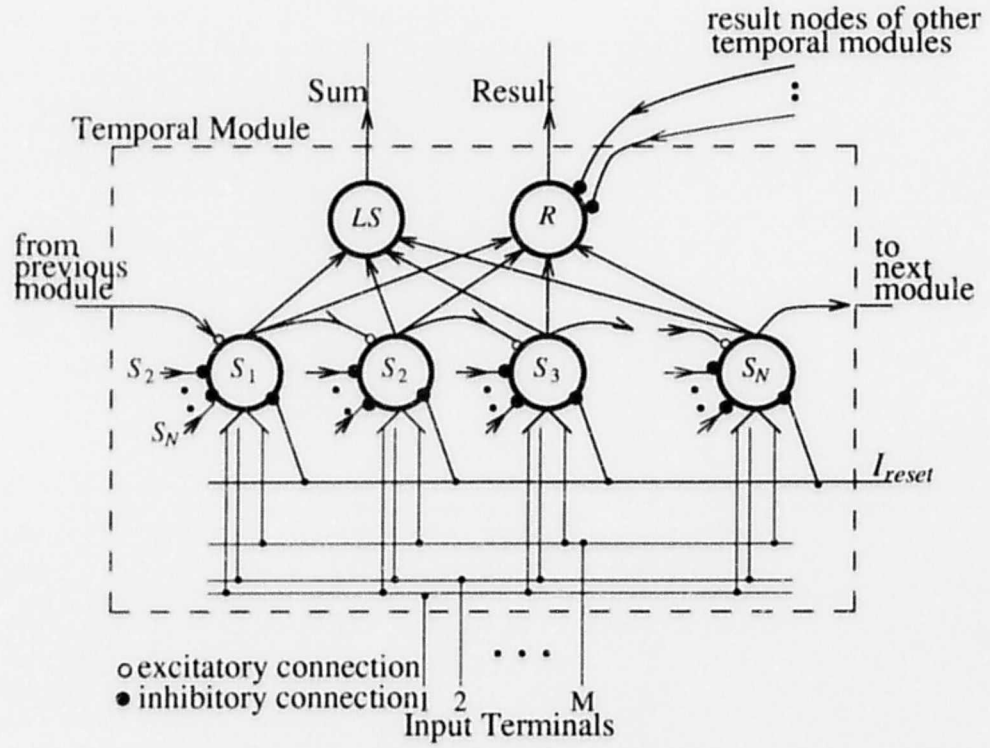



Fig. 6.1 A modified temporal module.

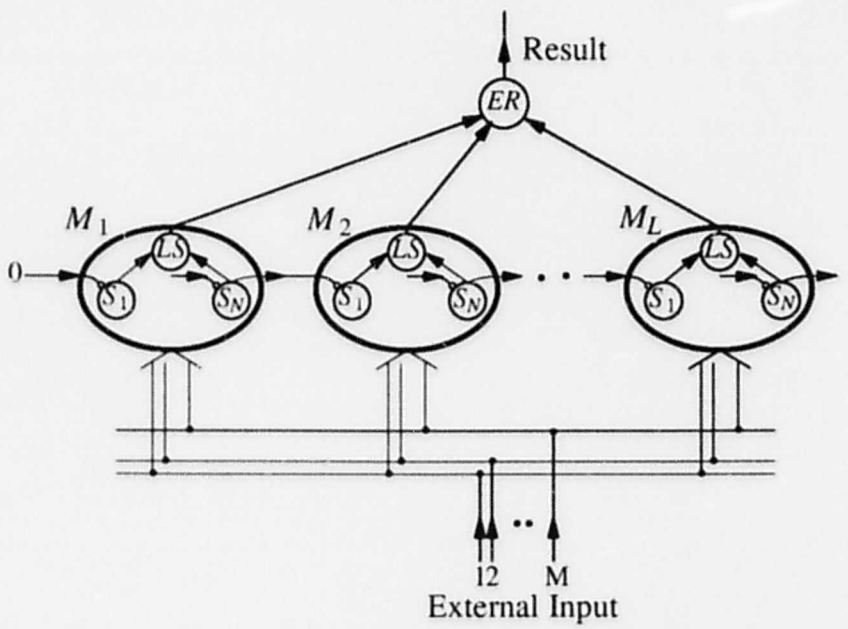



Fig. 6.2 A chain of temporal modules for recognition of a long sequence.

output of the linear sum node from each module is connected to the input of *ER* by a fixed-weight excitatory connection, and is temporally integrated by that node to generate an output representing the overall sequence. The external result node has a non-linear output response similar to the internal result node in each module, such that this node will only be activated if the overall sequence persists for a sufficient period. The integrator time constant in node *ER* should be selected based on the minimum length of the partial sequence that is to be recognized by the network.

The introduction of the linear sum node obviates the need for the internal result node in each module when the modules are cascaded in series. The internal result node may be used only to determine whether the subsequence corresponding to a particular model has been received. Of course, it is also needed when a module is used as a stand-alone unit.

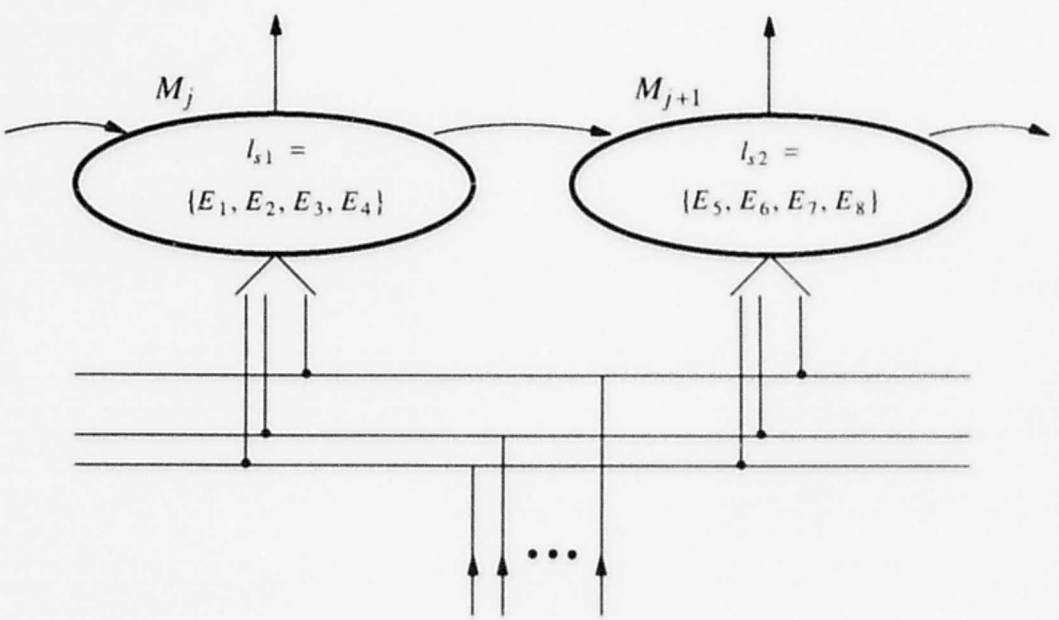



Fig. 6.3 A test sequence that overlaps two cascaded modules.

## 6.3. Hierarchical Interconnection

The idea of using a hierarchical structure as a means for reducing a network's complexity is not new [Simon 1962, Wolfram 1986]. For example, a hierarchical neural network was proposed by Fukushima for visual pattern recognition [Fukushima 1988]. Similarly, a hierarchical framework for vision was proposed by Tsotsos [Tsotsos 1988, Tsotsos 1989]. A number of temporal modules may be arranged as shown in Fig. 6.4, which shows a 2-level hierarchy. Each module in this figure may have the internal structure shown in Fig. 6.1 or may consist of a cascaded structure as in Fig. 6.2. The outputs of the result nodes of the modules on any level constitute the inputs to the modules on the next higher level.

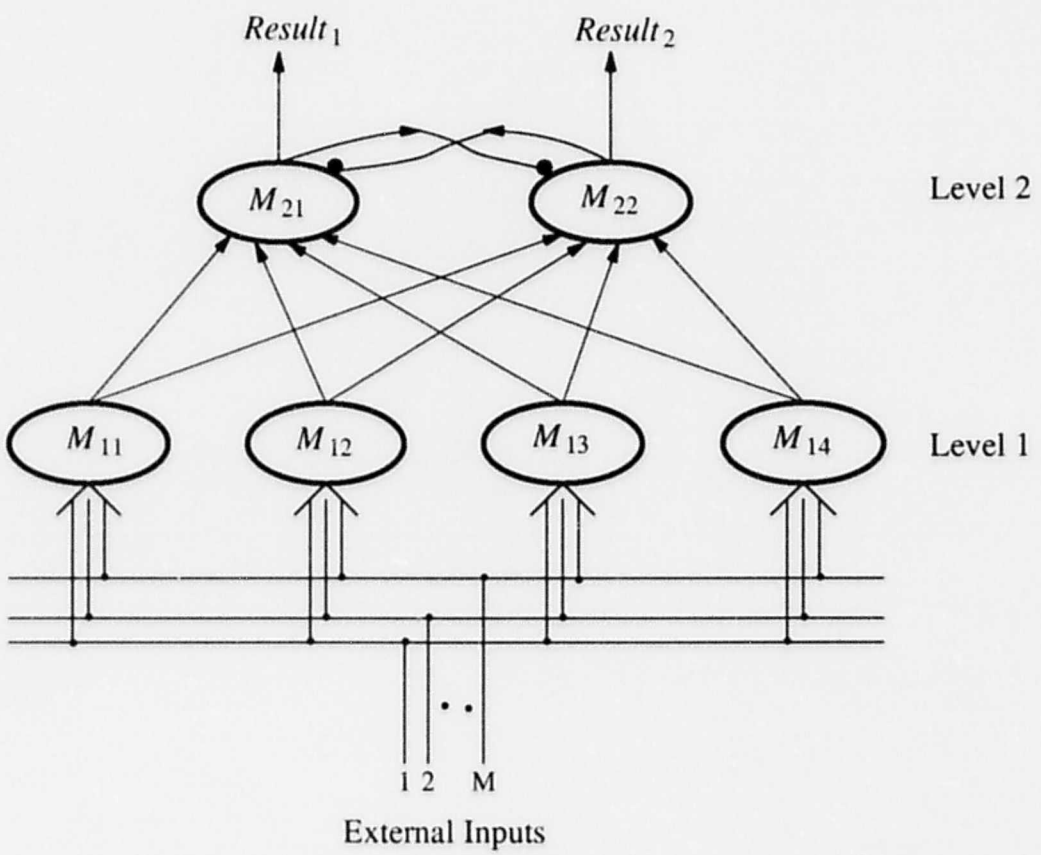



Fig. 6.4 A hierarchical neural network for temporal pattern recognition.

In a hierarchical network, the number of modules at any level is smaller than the number at the next lower level. Information is processed and abstracted by modules in one level before it is presented to modules in the next higher level, where it is further processed to obtain a higher form of representation. For ease of reference, we will call a sequence recognized by a module on level 1 a word. A sequence of words will be called a sentence. Hence, each module on level 2 can be trained to recognize an input sentence.

Modules on higher levels of a hierarchy must have larger time constants than modules at lower levels. This is necessary because the outputs of level-1 modules, for example, change at a slower rate than the external inputs. Modules on the highest level of the hierarchy receive inhibitory inputs from other modules on the same level, and operate in a winner-take-all fashion. The module recognizing an input sentence achieves the maximum output, while outputs of all the other modules on the same level are inhibited.

Because of the non-linearity of the result nodes, a module may be activated by a partial sequence, provided the sequence is sufficiently long to exceed the node's activation level. This means that a module on level 1 can recognize a word even when a part of that word is missing. Similarly, a module on level 2 can recognize a sentence in which some words are missing. The extent of this tolerance can be controlled by proper selection of the time constants and biases of the result nodes.

To minimize the number of modules required, it is possible to have the modules on any level of hierarchy to be shared by the modules on the next higher level. For example, if modules $M_{11}$, $M_{12}$, $M_{13}$, and $M_{14}$ are trained to recognize words $l_{s1}$, $l_{s2}$, $l_{s3}$, and $l_{s4}$ respectively, then module $M_{21}$ can be trained to recognize sentence $\{l_{s1}, l_{s2}, l_{s3}\}$ and $M_{22}$ can be trained to recognize sentence $\{l_{s1}, l_{s2}, l_{s4}\}$. Modules $M_{11}$ and $M_{12}$ are shared by modules $M_{21}$ and $M_{22}$ on the next level of hierarchy.

## 6.4. Simulation Results

The operation of the enhanced temporal modules, which can be connected either in series or in a hierarchical structure, has been tested by implementing a motion detector similar to that described in chapter 5. This section presents the test results, which have been obtained by simulation.

### 6.4.1. Series Connection

A motion detector with three concatenated modules, each of which comprises ten sequence nodes, has been simulated and tested. Modules $M_1$, $M_2$ and $M_3$ were independently trained to recognize sequences $l_{s1}$, $l_{s2}$ and $l_{s3}$, respectively, where each sequence consists of 10 events representing the movement of an object. During training, the weights associated with inter-node connections and the internal bias in each node are adjusted using the LMS algorithm presented in chapters 3 and 4. The external result node in Fig. 6.2 was trained to recognize the concatenated sequence $\{l_{s1}, l_{s2}, l_{s3}\}$ by presenting the actual sequence.

The detector is tested using test sequence $t_{si}$ where $D(l_{si}, t_{si}) = 0$, $i = 1, 2, 3$. The responses of the linear sum node in each module and the external result node are shown in Figs. 6.5 and 6.6 for two input sequences. In the case of Fig. 6.5, a random sequence $t_{x1}$ was first presented to the machine, followed by $t_{s1p}$, which is a partial sequence taken from the later part of $t_{s1}$. Sequence $t_2$ was then presented to the machine, followed by another partial sequence, $t_{3p}$, taken from the early part of $t_3$. Finally, another random sequence, $t_{x2}$, was presented. The figure shows that the output of the linear sum node of module $M_1$ rises partially, followed by the complete activation of $M_2$ and partial activation of $M_3$. The temporal integration of the activities at the linear sum nodes causes the external result node to be activated gradually.

Fig. 6.6 depicts the case of a slightly modified input sequence. An early part of sequence $t_{s1}$, $t'_{s1p}$, was presented to the machine, followed by a random sequence, $t_{x1}$. In response, the output of the linear sum node of module $M_1$ rises gradually to 0.5 then decays to 0 at time $(t_0 + 10T)$. When

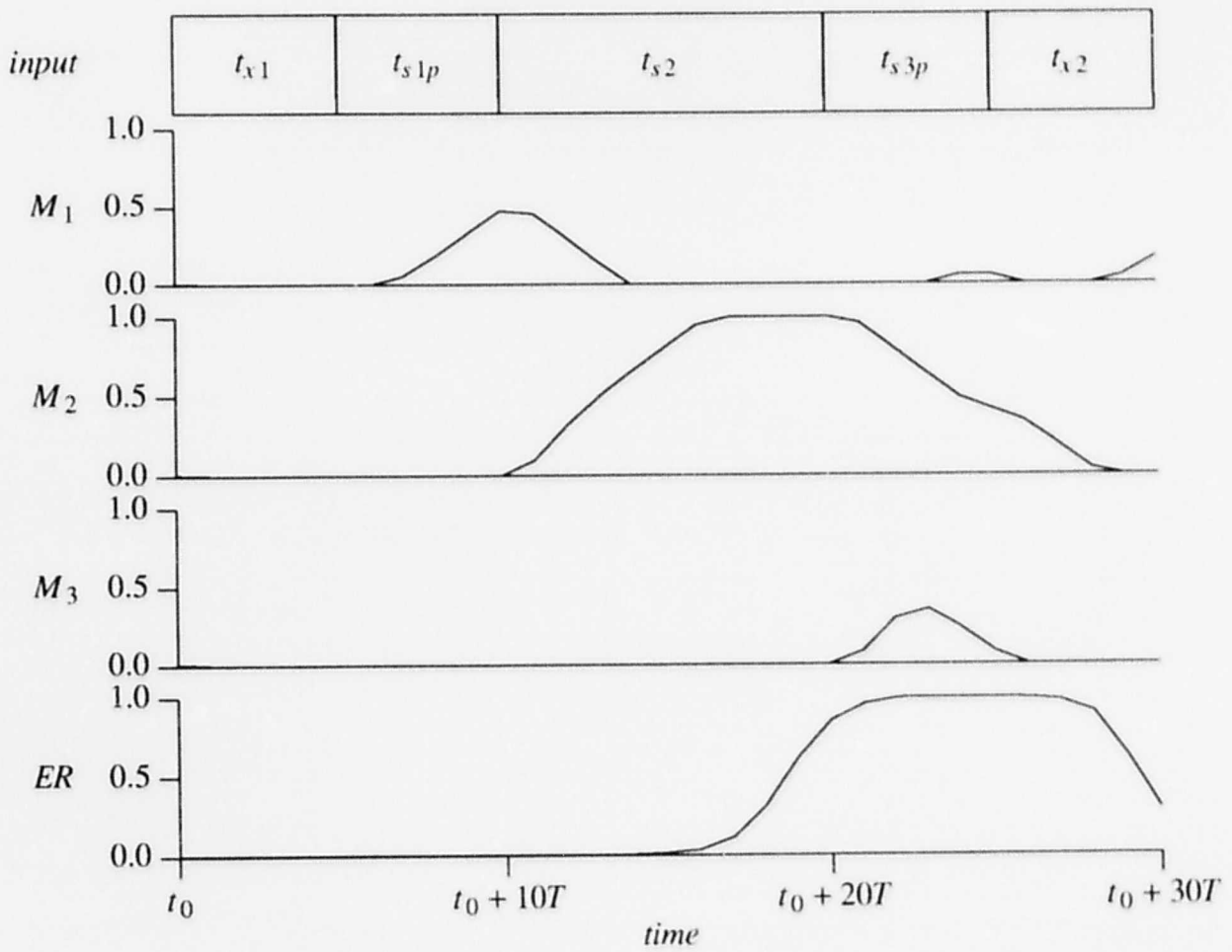



Fig. 6.5 Outputs of the linear sum nodes of modules $M_1$, $M_2$, $M_3$, and $ER$ when sequences $t_{x1}$, $t_{s1p}$, $t_{s2}$, $t_{s3p}$, and $t_{x2}$ are presented to the network.

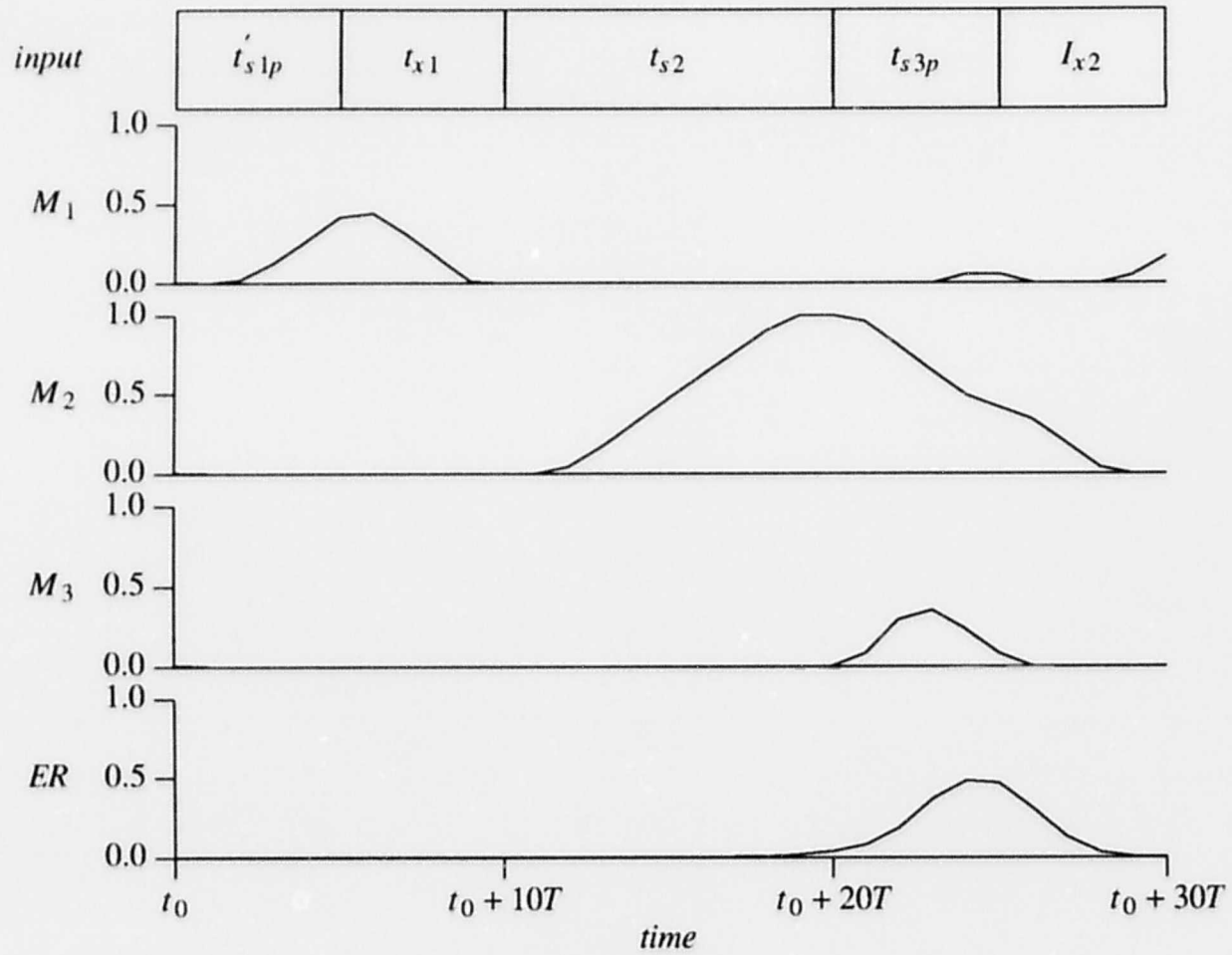



Fig. 6.6 Outputs of the linear sum nodes of modules $M_1$, $M_2$, $M_3$, and $ER$ when sequences $t_{s1p}$, $t_{x1}$, $t_{s2}$, $t_{s3p}$, and $t_{x2}$ are presented to the network.

sequence $t_{s2}$ is presented, module $M_2$ is activated later than in Fig. 6.5, because its first sequence node does not receive an excitation signal from module $M_1$. The slower activation of modules $M_2$ and the fact that the response of $M_1$ has been shifted to the left causes the external result node to be weakly activated, demonstrating that disjoint sequences are not sufficient to activate the external result node.

### 6.4.2. Hierarchical Network

A two-level hierarchical neural network of the type shown in Fig. 6.4 has been used to implement a motion detector. Modules $M_{11}$, $M_{12}$, $M_{13}$ and $M_{14}$ were trained independently to recognize words $l_{s1}$, $l_{s2}$, $l_{s3}$ and $l_{s4}$, respectively. Each word consists of 6 events depicting a moving object. After the modules in level-1 have been trained successfully, modules $M_{21}$ and $M_{22}$ on level 2 were then trained to recognize the sentences $\{l_{s1}, l_{s2}, l_{s3}\}$ and $\{l_{s1}, l_{s2}, l_{s4}\}$, respectively.

The hierarchical network is tested using a sentence that consists of 3 words. Word $t_{si}$ is selected such that $D\ (l_{si}, t_{si}) = 0, i = 1, .. 4$. Fig. 6.7 shows the outputs of various modules when the test sentence $\{t_{s1}, t_{s2}, t_{s3}\}$ is presented. Modules $M_{11}$, $M_{12}$, and $M_{13}$ are activated in order by their respective sequences, while module $M_{14}$ remains reset. By integrating the activities of these modules, module $M_{21}$ becomes active at time $(t_0 + 15T)$, indicating that it recognizes this input correctly as one complete sequence. Module $M_{22}$ begins to fire when $t_{s1}$ and $t_{s2}$ are presented, but is later reset because module $M_{14}$ did not fire. In another test (not shown), when the sequence $\{t_{s1}, t_{s2}, t_{s4}\}$ was presented, module $M_{22}$ was activated while module $M_{21}$ remained quiescent.

In order to demonstrate that the modules in the higher level of the hierarchy can abstract the temporal order of the sequences recognized by the modules on a lower level, an input sequence that has $t_{s1}$, $t_{s2}$ and $t_{s3}$ arranged in reverse temporal order was presented. Fig. 6.8 depicts the modules' responses. Modules $M_{21}$ and $M_{22}$ on level 2 remain quiet, even though the modules on level 1 respond to their respective sequences. This clearly indicates that $M_{21}$ is only responsive when the words of a sentence are presented in the correct temporal order.

Consider now the responses of the modules in the upper level when only partial words and partial sentences are presented to the network. Fig. 6.9 shows that the network continues to function properly when only parts of $t_{s2}$ and $t_{s3}$, separated by a random sequence, are presented, provided that these parts are sufficiently long to be recognized by their respective modules.

A network consisting of a single level of modules requires the same number of sequence nodes as the total number of events in the sequences to be recognized. The hierarchical network

presented here clearly illustrates that the number of sequence nodes can be reduced by having the modules in a higher level of the hierarchy share some of the modules in the lower levels. In the example given, modules $M_{11}$ and $M_{12}$ on level 1 are shared by modules $M_{21}$ and $M_{22}$ on level 2.

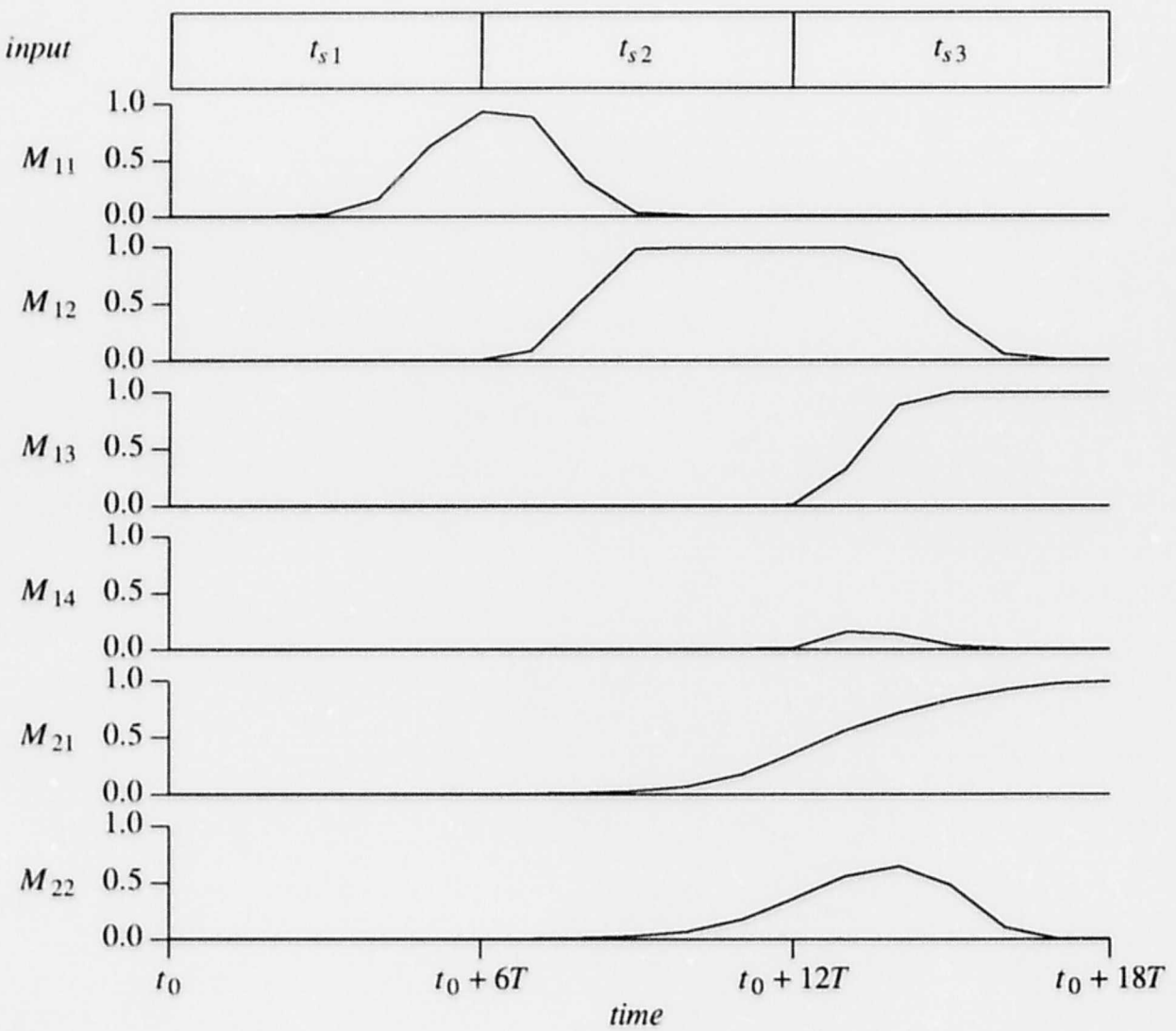



Fig. 6.7 Outputs of modules $M_{11}$, $M_{12}$, $M_{13}$ $M_{14}$, $M_{21}$, and $M_{22}$ when sequences $t_{s1}$, $t_{s2}$, and $t_{s3}$ are presented to the network.

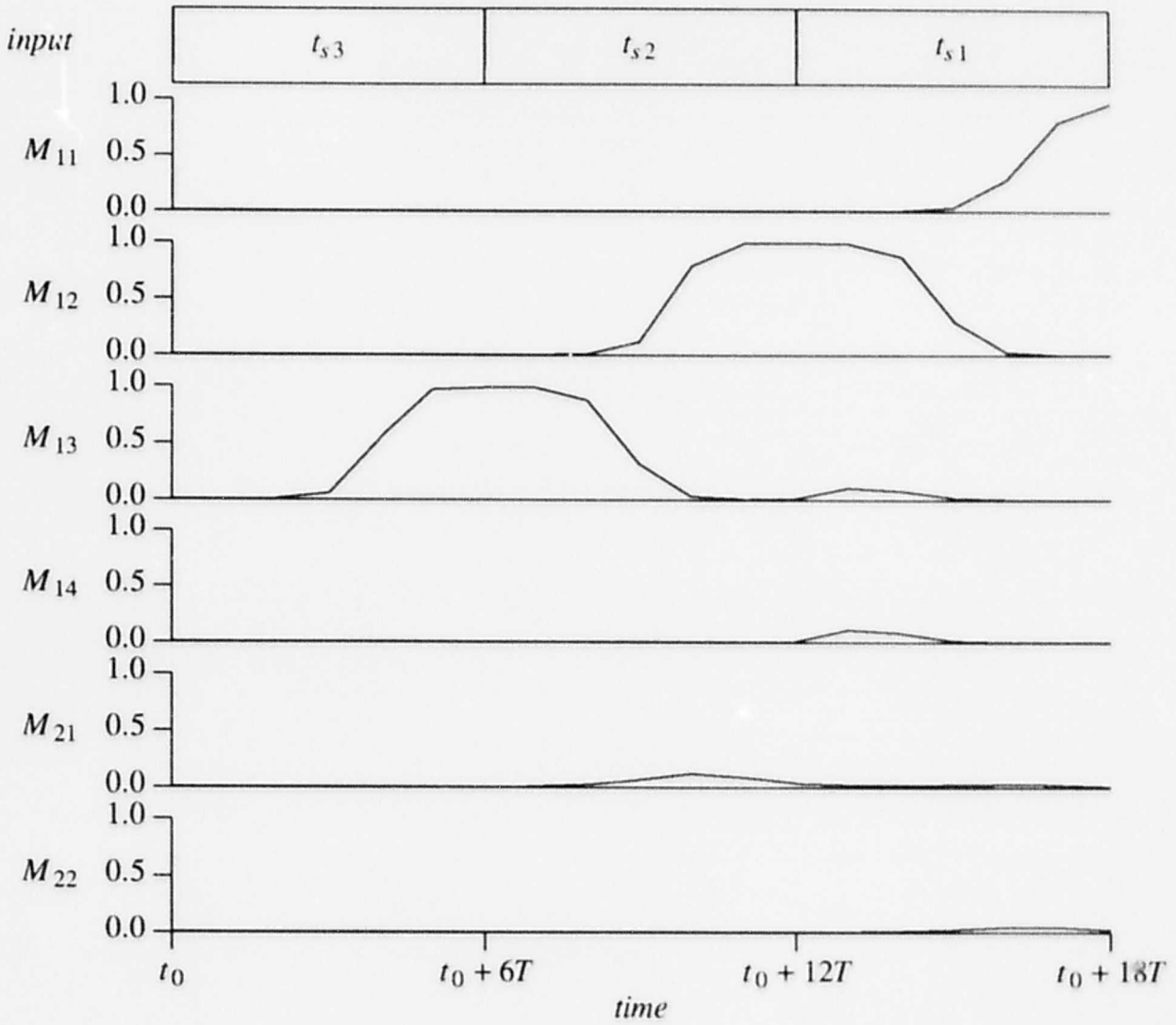



Fig. 6.8 Outputs of modules $M_{11}$, $M_{12}$, $M_{13}$ $M_{14}$, $M_{21}$, and $M_{22}$ when sequences $t_{s3}$, $t_{s2}$, and $t_{s1}$ are presented to the network.

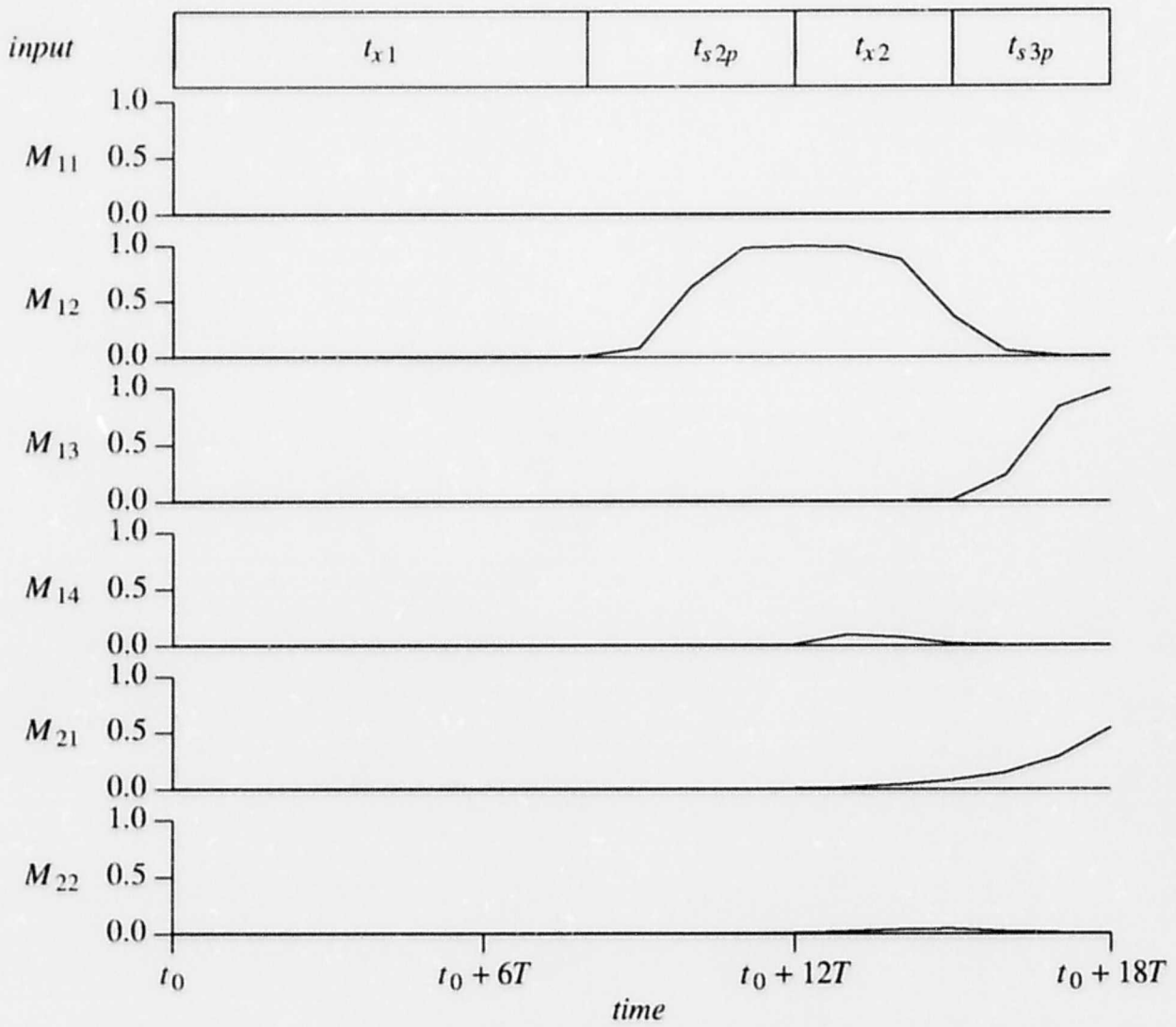



Fig. 6.9 Outputs of modules $M_{11}$, $M_{12}$, $M_{13}$ $M_{14}$, $M_{21}$, and $M_{22}$ when sequences $t_{x1}$, $t_{s2p}$, $t_{x2}$ and $t_{s3p}$ are presented to the network.

## 6.5. Conclusion

This chapter presented an enhanced neural network module suited for implementation on a single VLSI chip. A number of such modules can be interconnected to recognize temporal sequences of varying length. Two interconnection schemes have been investigated. A series connection combined with linear sum nodes enables the modules to recognize long sequences and par-

tial sequences that may overlap module boundaries. A hierarchical structure yields a network that can recognize the temporal order of several partial sequences.

# Chapter 7

# VLSI Implementation

## 7.1. Introduction

This chapter discusses the implementation of the temporal module introduced in Chapter 6 in the form of a VLSI chip. The purpose of the chapter is to investigate the feasibility of implementing various functional elements and to discuss the difficulties and trade-off involved.

Some existing VLSI implementations of neural networks are presented first, followed by circuit models for the building blocks of a temporal module. Operation of the proposed circuits have been verified using the SPICE circuit simulator, and the results of simulation are presented.

## 7.2. Existing Implementations of a Neural Network

The developments in artificial neural networks have focussed on software simulations, with emphasis on hardware implementations appearing only recently [Mead 1980, Mead 1989]. A neural network can be integrated onto a chip, where nodes are implemented as a collection of connected analog or digital circuits. Because parallel computations are possible on a chip, the VLSI implementation of a neural network provides the capability to process the incoming information in real time.

There are two possible approaches to the implementation of a neural network on a chip. In the first approach, analog VLSI circuits such as transconductance amplifiers are used as the basic building blocks. The second approach uses conventional digital circuits to implement a network.

## 7.2.1. Analog Implementation

This is a popular approach for the implementation of a neural network on a chip. Some examples of recent work can be found in [Sivilotti 1986, Graf 1988, Mead 1988, Alspector 1989, Boahen 1989, Graf 1989, Lazarro 1989, Maher 1989, Mead 1989, Reed 1989, Verleysen 1989]. Consider the network node shown in Fig. 7.1, which consists of inputs with weighted connections, a summer, and an amplifier with a non-linear output. The following presents briefly an analog implementation of these components.

The input to a summer of one node is connected to the output of another node by a MOS (Metal Oxide Semiconductor) transistor, which implements a weighted interconnection. The gate which controls the channel conductance of the transistor, hence the connection strength, is connected to a voltage source. Each node has a non-inverting output, $v_j$, and an inverting output $\bar{v}_j$. A

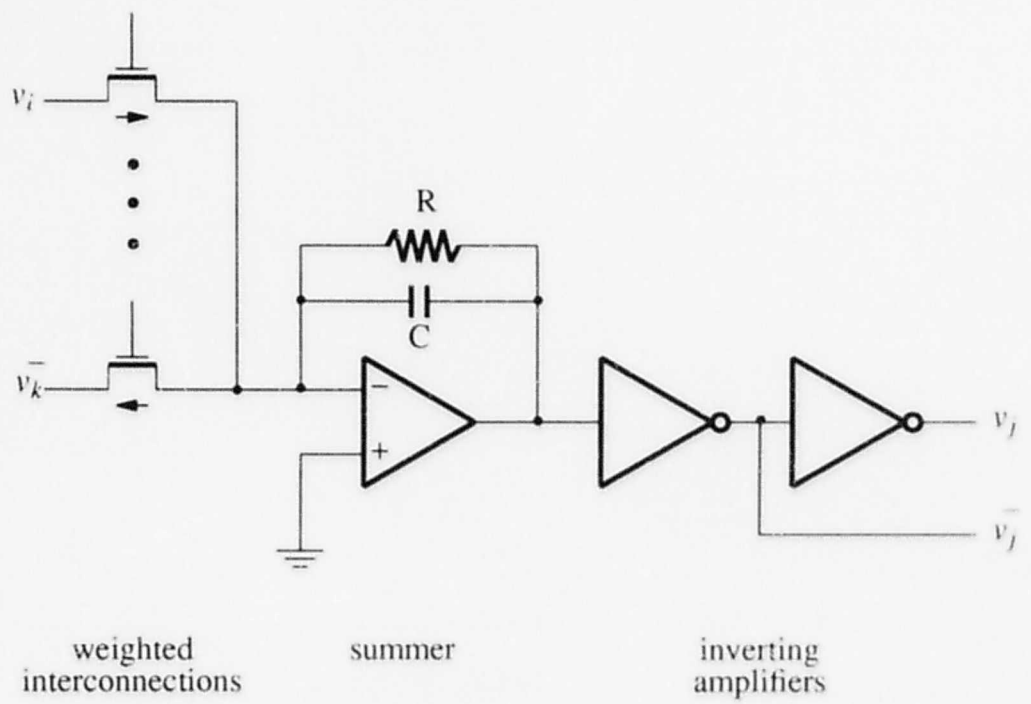



Fig. 7.1 A circuit implementing the node of a neural network.

positive weight is implemented by connecting the drain of the MOS transistor to the non-inverted output of another node, and to the inverted output for a negative weight. Current is injected into a node when the weight is positive and vice-versa for a negative weight. Interconnect weights can be stored in a programmable memory which determines the voltage applied to the gates of the input transistors. Alternatively, the weights can be stored in an analog fashion by replacing the MOS transistor connecting two nodes by a floating-gate MOS transistor similar to those used in programmable read-only memories. With this technique, the channel conductance is determined by the charge stored on the floating gate. This type of storage is nonvolatile. The weights have to be computed externally and then loaded onto the chip.

The summer is implemented by using an operation amplifier configured as a leaky integrator to sum all incoming currents injected into a node. The output of the summer is connected to the input of an inverting amplifier which implements the required non-linear transfer function of a node when saturation of the amplifier is taken into account.

### 7.2.2. Digital Implementation

Three approaches to implement a neural network based on digital VLSI technique have been reported in the literature. First, a neural network can be mapped onto a multi-processor, where each processor is responsible for the computations involving only a few nodes and the input is distributed to all processors by a host computer. The result from each of these processors is retrieved and combined by the host computer to generate the final output. The performance of neural networks mapped onto a shared-memory, ring, and mesh multi-processor topologies were compared and presented in [Kung 1988, Suzuki 1989, Zitti 1989].

A high speed floating-point multiplier in a digital signal processor (DSP) leads to excellent performance in computing inner products such as those in Eqn. 2.2. As a result, another approach to the digital implementation of a neural network is to use a single DSP chip to perform the computations for all the network nodes. This technique has been successfully applied to recognize written postal digits in real time [LeCun 1989b]. To utilize the potential parallelism available in a neural

network, several DSPs can be used to perform the necessary computations in parallel by partitioning a job among them.

The third approach to a digital implementation of a neural network is based on pulse-density modulation, implemented by using switched-capacitor structures as shown in Fig. 7.2 [Tomberg 1989, Habib 1989, Van Den Bout 1989]. The output of a node is encoded as a stream of pulses, where the number of pulses per unit time represents the output strength. The weighted input to a node is computed using an exclusive-or logic gate to multiply the pulse-density coded output from another node by the pulse-density coded weight stored in a ring shift register. A switched capacitor circuit sums all the weighted inputs and generates an output which, in turn, drives the input of another node.

## 7.3. Analog Implementation of Temporal Module

This section presents an analog implementation of a temporal module. The circuit model and the circuit design of each building block for a node are presented. Operation of the proposed circuits have been verified using the SPICE circuit simulator, as described below.

### 7.3.1. Circuit Model

A temporal module consists of several nodes, each of which consists of a non-linear output element, a fading memory, and several weighted inputs. These nodes may be implemented using the circuit shown in Fig. 7.3, which comprises a voltage amplifier fed by a number of voltage-controlled current sources. The capacitor and resistor at the input of the amplifier provides the fading memory, whose time constant is given by $\tau = \frac{C}{G}$. The input of the amplifier is connected to the outputs of other nodes via the voltage-dependent current sources, which implement the necessary excitatory or inhibitory weighted connections. For an excitatory connection, a current source injects current into a node whereas it withdraws current from the node in the case of an inhibitory connection. The weight of the connection is defined by the mutual conductance of the current

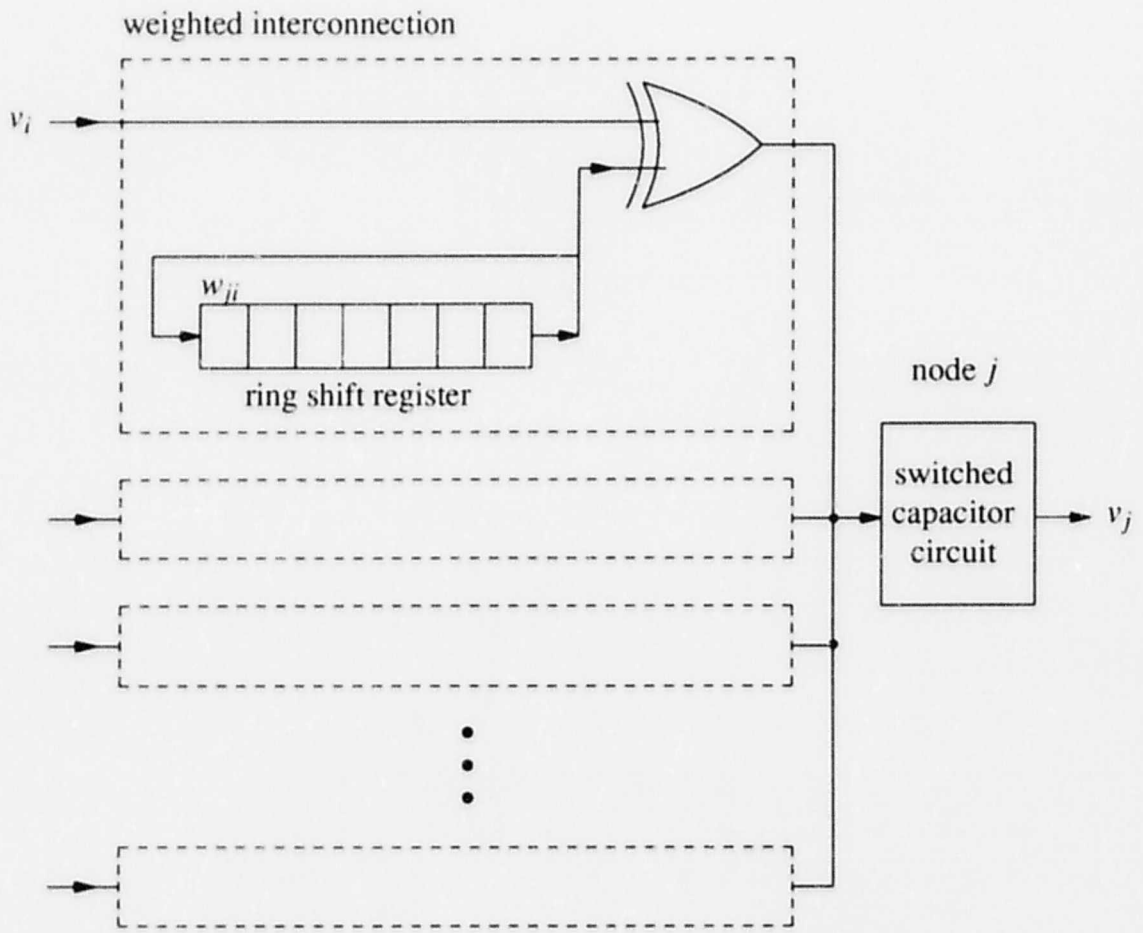



Fig. 7.2 An interconnect for a switched-capacitor node.

source.

The internal state of each node proposed in chapter 3 changes with a fixed time constant satisfying a first order linear differential equation given by Eqn. 3.3. The same form as Eqn. 3.3 can be derived for the circuit in Fig. 7.3. Applying Kirchoff's current Law at the input of the amplifier, we obtain:

$$\tau \frac{du_j}{dt} + u_j = \frac{1}{G}\left( \sum_{i=1, i \neq j}^{N} v_i g_{ji} + V_{DD} f_j \right) \tag{7.1}$$

where $u_j$ is the voltage at the input of the amplifier of node $j$, $v_i$ is the output of another node $i$, $g_{ji}$

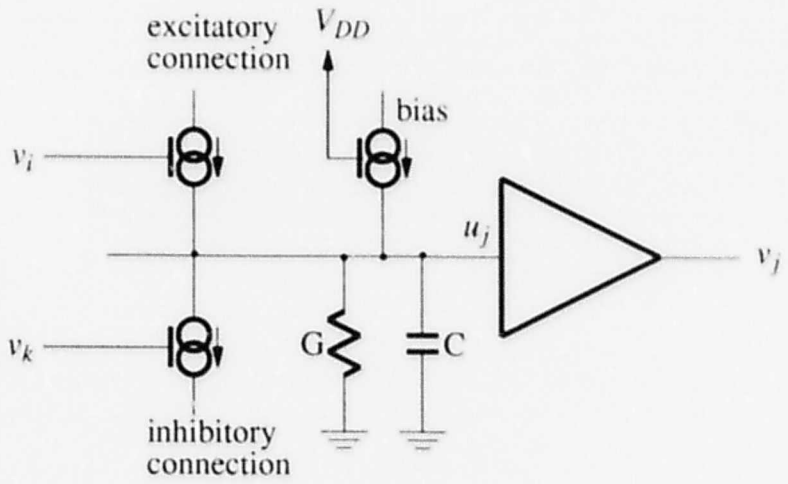



Fig. 7.3 A circuit model of a node.

is the mutual conductance of the current source connecting node $i$ to node $j$, and $f_j$ is the mutual conductance of the current source defining the bias of node $j$.

The next section presents the VLSI circuits that are proposed for various components. The purpose is mainly to demonstrate the feasibility of implementing a temporal module on a VLSI chip. No detailed study of performance has been conducted and the circuits have not been built.

## 7.3.2. Components' VLSI Circuits

VLSI circuits for the components in Fig. 7.3 are presented in this section. The simulation results given have been obtained using the SPICE circuit simulator [Vladimirescu 1981].

### 7.3.2.1. Voltage Amplifier

All nodes, except the linear sum node (*LS*) presented in chapter 6, have a non-linear output element, which can be implemented by a voltage amplifier. A voltage amplifier usually behaves linearly for a certain input range, and enters saturation outside that. This behavior is closely approximated by a piecewise linear transfer function, which in turn is a suitable approximation for

a sigmoidal function.

A possible implementation for a voltage amplifier is shown in Fig. 7.4. The circuit consists of a differential pair, $Q_1 - Q_2$, a bias transistor, $Q_b$, and a pullup resistor, R. The bias transistor, $Q_b$, is used as a current source by operating the transistor in its saturated region. The bias current controls the amount of current flowing through the differential pair, thus controlling the voltage gain. When the input voltage $v_{in}$ is less than 0, current $i_1$ is close to 0, and $i_2$ is close to the bias current $i_b$. Therefore, the output voltage is equal to $(V_{DD} - Ri_b)$. When $v_{in}$ is greater than 0, current $i_1$ is close to $i_b$, and $i_2$ is close to 0, and the output voltage is equal to $V_{DD}$. When $v_{in}$ is equal to 0, currents $i_1$ and $i_2$ are both equal to $\frac{i_b}{2}$, causing the output voltage to be equal to $(V_{DD} - \frac{Ri_b}{2})$.

The circuit presented above requires a high load resistance $R$, which takes up quite a large area on the chip. This would mean a lesser number of devices that can be integrated on a chip. A better approach is to use a saturated NMOS enhancement transistor, as shown in Fig. 7.5. The

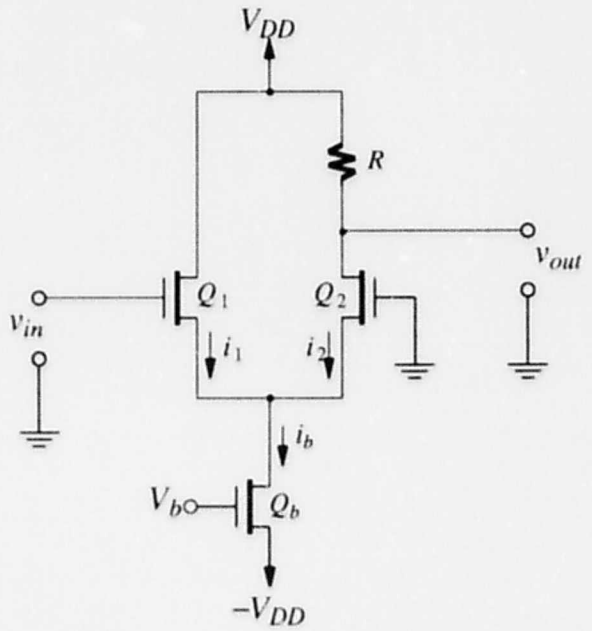



Fig. 7.4 A circuit diagram of a voltage amplifier using a resistive load.

transistor offers a high resistance, because it is operating in the saturated mode, and requires a smaller area. In spite of the advantage, there is a limitation for using this type of load, which is that the output voltage never reaches $V_{DD}$. However, as we shall see in the next section, this limitation is not a hindrance to the design.

The circuit in Fig. 7.5 was simulated using SPICE, and the output of the circuit for different values of input is shown as a solid line in Fig. 7.6. The supply and bias voltages used in the simulation are $V_{DD} = 5$ volts and $V_b = -3.5$ volts, respectively. Because $Q_3$ is operating in the enhancement mode, the output varies from 0.4 to 1.8 volts.

For comparison, the sigmoidal function used in the simulations presented in Chapter 3 is shown as a dotted line in the Fig. 7.6. Except for the 0.4 volts of output voltage offset, the input-output characteristic of the amplifier is similar to the sigmoidal function. We shall see in the next section that this offset will not adversely affect the operation of a voltage amplifier as the non-linear output element in a node.

The series connection of modules proposed in chapter 6 requires the use of a linear sum node in each module. A linear sum node has a linear output element, which can be difficult to implement. However, a linear function of the node can be approximated by a voltage amplifier that has a wide linear input range. This can be achieved by applying a smaller bias voltage $V_b$.

### 7.3.2.2. Excitatory Interconnections

An excitatory connection is implemented by a voltage-dependent current source that injects current into the node. A circuit for the current source is shown in Fig. 7.7. It consists of a current mirror, $Q_1 - Q_2$, and a bias transistor, $Q_b$, connected to a fixed bias voltage, $V_b$. The bias transistor functions as a voltage-dependent current source, and current $i_b$ is proportional to both the input voltage and the geometric ratio $(\frac{W}{L})$ of the transistor, as given in the following

$$i_b = K(\frac{W}{L})(2(V_{GS} - V_T) - V_{DS})V_{DS} \tag{7.2}$$

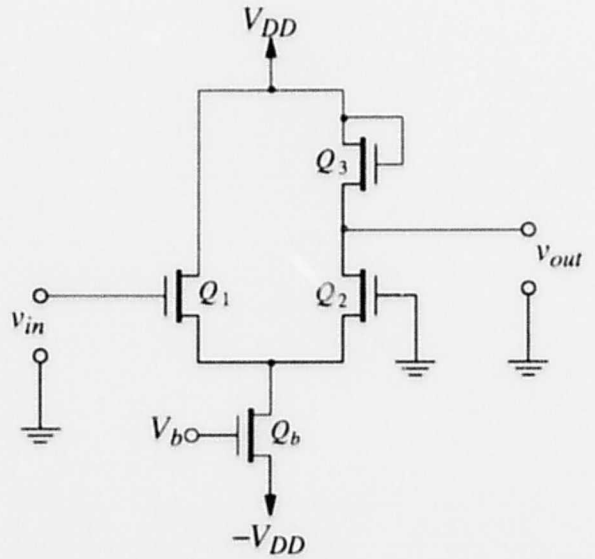



Fig. 7.5 A circuit diagram of a voltage amplifier.

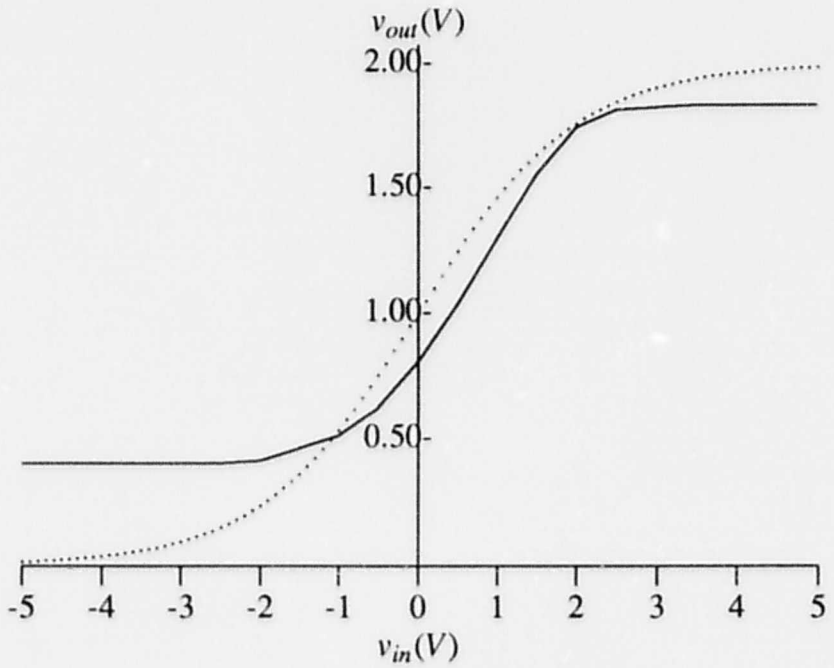



Fig. 7.6 Plots of $v_{out}$ versus $v_{in}$ for the circuit of Fig. 7.5 (solid line) and the sigmoidal function (dotted line).

where $V_{GS} = v_{in} - V_b$, $K$ is a circuit constant, $W$ and $L$ are the channel width and length of the bias transistor $Q_b$, and $V_{GS}$, $V_T$ and $V_{DS}$ are the gate-to-source voltage, the threshold voltage, and the drain-to-source voltage of the bias transistor, respectively [Mead 1980, Mead 1989]. Note that this equation only applies when the bias transistor operates in the ohmic region. Using the current mirror provided by transistors $Q_1 - Q_2$, current $i_0$ injected into a node is equal to current $i_b$ of the bias transistor.

Results of SPICE simulation of this circuit are shown in Fig. 7.8. The supply and bias voltages are $V_{DD} = 5$ volts and $V_b = -1.8$ volts, respectively. The performance of the circuit using bias transistors with different geometric ratios is shown. When $\frac{W}{L} = 1$, the circuit is approximately linear between 0.8 and 1.6 volts, and the circuit saturates with a current of 44 $\mu A$ for input voltages greater than 3 volts. When the $\frac{W}{L}$ ratio is doubled, the circuit is approximately linear between 0.8 and 1.2 volts, and saturates with an output current of 46 $\mu A$ when the input voltage is greater than 2 volts.

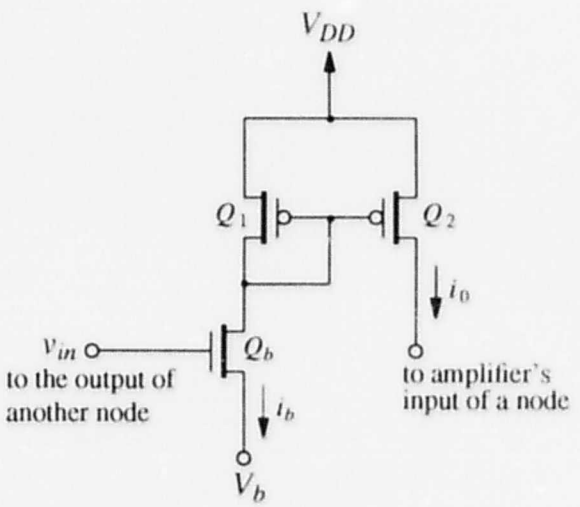



Fig. 7.7 A circuit diagram of an excitatory connection

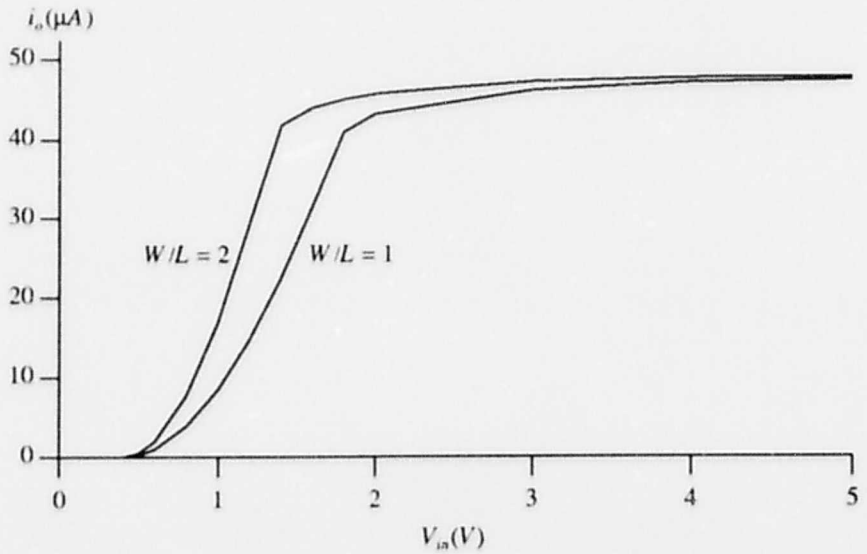



Fig. 7.8 Plot of $i_0$ versus $v_{in}$ of an excitatory connection.

Fig. 7.6 and 7.8 demonstrate that the circuit in Fig. 7.7 can implement the function of a linear excitatory connection. The voltage amplifier has its output voltage level between 0.4 and 1.8 volts, which approximately matches the linear range of input voltage for the circuit in Fig. 7.7.

The weight of a connection can be realized by using bias transistors of a different geometric ratio. A stronger weight requires a bias transistor having a higher $\frac{W}{L}$ ratio. This approach of implementing a weighted interconnect on a chip leads to fixed weights for a given chip. That is, for a specified application, interconnect weights would be predetermined and programmed onto the chip using bias transistors of different sizes. Several VLSI implementations of neural networks that have been tailored for special applications have been reported by Mead [Mead 1989].

The interconnect weight can be made programmable by modifying the interconnection circuit as shown in Fig. 7.9. The figure shows a circuit with a two-bit memory. There are four different values of transconductance controlled by the memory cells. Each cell is connected to the gate of a

pass transistor. When a 1 is stored in a memory cell the corresponding pass transistor is switched on, allowing the current to pass through. The pass transistor is switched off otherwise.

The resolution of the weight of the interconnection implemented using the above approach is limited by the number of transistors needed. High resolution requires a large chip area, which means that fewer nodes can be implemented on a chip. Future work is required to determine the minimum level of resolution needed for the interconnect weights in a given application.

### 7.3.2.3. Inhibitory Interconnections

An inhibitory connection consists of a voltage-dependent current sink that withdraws current away from a node. It can be implemented by adding another current mirror, $Q_3 - Q_4$, to the circuit shown in Fig. 7.7, as shown in Fig. 7.10. This device has the same transconductance characteristics as shown in Fig. 7.8.

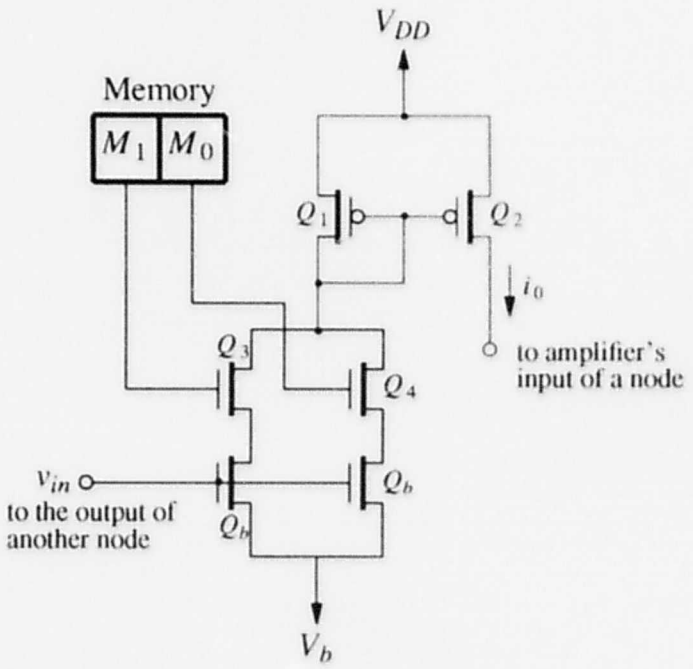



Fig. 7.9 An excitatory connection with selectable transconductance.

### 7.3.2.4. Generalized Interconnections

The interconnections between the nodes in a temporal module are either excitatory or inhibitory connections. They can be implemented by choosing one of the two circuits in Figs. 7.7 and 7.10. However, all the sequence nodes are also linked to the external inputs by weighted connections, which may be either excitatory or inhibitory. Hence, it is useful to have a generalized interconnection that can function either as an excitatory or inhibitory connection.

A possible design for an interconnection is illustrated in Fig. 7.11. A memory bit is used to remember whether the connection is excitatory or inhibitory, and two bits, $M_1$ and $M_0$, to remember the interconnect weight.

When bit $S$ is 1, pass transistors $Q_{p1}$ and $Q_{p3}$ are switched off and $Q_{p2}$ is switched on. As a result, the current mirror consisting of $Q_5$–$Q_6$ is inactive, and $i_0$ is injected into the input of the amplifier. When bit $S$ is 0, pass transistor $Q_{p2}$ is switched off while the other two pass transistors are switched on. This causes current $i_0$ to be injected into the current mirror, $Q_5$–$Q_6$, via pass

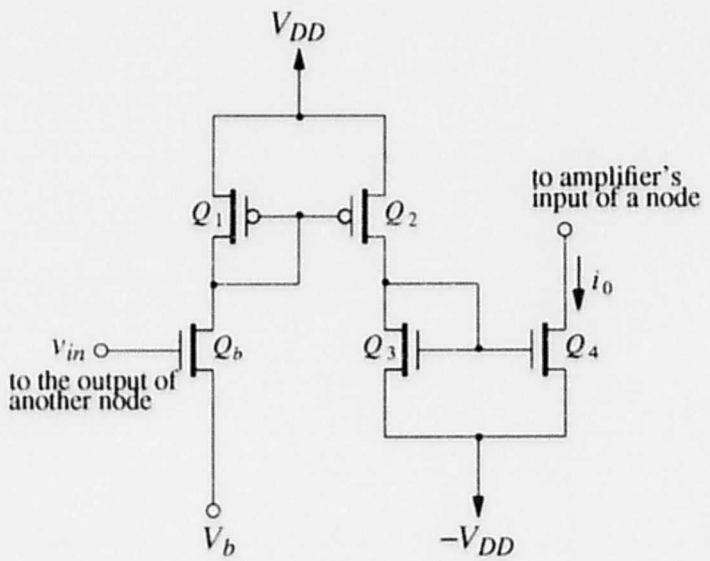



Fig. 7.10 A circuit diagram of an inhibitory connection

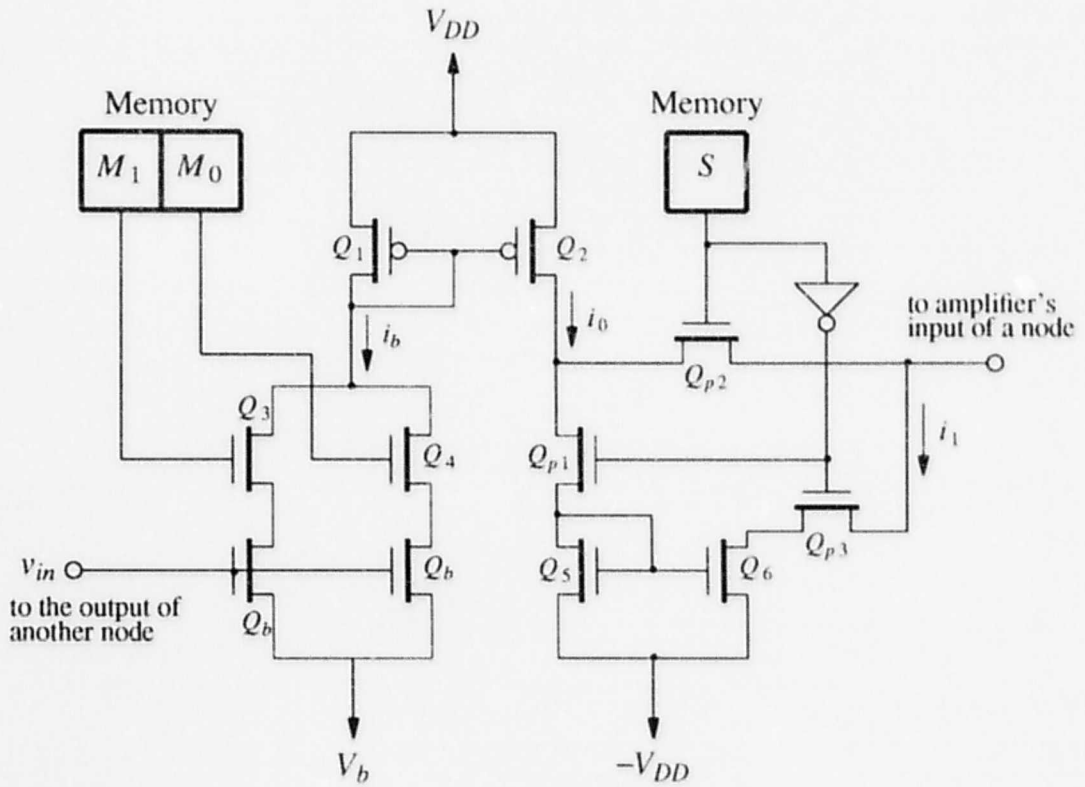



Fig. 7.11 A generalized interconnection.

transistor $Q_{p1}$. As a result, $i_1$, which is equal to $i_b$, will be drawn from the node via the pass transistor $Q_{p3}$.

## 7.4. Conclusion

This chapter has demonstrated the feasibility of implementing the proposed temporal module on a VLSI chip. A circuit model of a node shows that there is a VLSI equivalent of the module's nodes and interconnections, and suitable circuits have been proposed for each of the functional elements in that model.

# Chapter 8

# Conclusions and Future Work

## 8.1. Conclusions

The main thesis of the work is that a special configuration of a sequential neural network that uses a separate memory element for each state along its state trajectory is well suited for use in temporal pattern recognition. The network state that represents the recognition results of individual events in a temporal sequence can be observed easily, and thus can be used to generate a final response through a process of temporal integration.

The combined use of temporal integration and one memory element per state yields a sequential neural network that can recognize partial sequences, and is tolerant to noise. Because of the use of one memory element per state, the network has a simple topology, with no hidden nodes. All the intermediate states of the network can be enumerated, which allows the network to be trained easily using the LMS algorithm. Simplified analysis of the network's behavior is also possible. The analysis allows a better understanding of the network dynamics and enables the network parameters to be estimated.

The capabilities of the network structure in this thesis can be expanded in a modular, hierarchical fashion. A number of modules, each trained to recognize a particular temporal sequence, can be used for recognition of multiple sequences. Modules can be connected in series to recognize a long temporal sequence, or in a hierarchical structure to recognize a sequence of sequences. Simulation results show that a hierarchical temporal network has the ability of abstracting the temporal order of a number of input sequences, where each may be a partial sequence.

The network was evaluated by simulating its use to implement a motion detector. With a moving random-dot pattern as the test input, a motion detector using several temporal modules has been shown to recognize a variety of input sequences resulting from the motion of an object along a certain trajectory at different speeds. Results indicate that a trained detector can recognize the motion of objects of varying sizes along a trajectory and can recognize partial motion sequences. The detector is tolerant to intermittent loss of the input image and degrades gracefully with increasing input noise.

The proposed temporal module requires a small number of external connections. Hence, it is well suited for implementation in the form of a VLSI chip. Several MOS circuits have been described to illustrate this feasibility.

## 8.2. Thesis Contributions

The contributions of this thesis are summarized below:

- Identification of a special configuration of a sequential neural network that is well suited for temporal pattern recognition.
- Piecewise-linear analysis of the proposed neural network.
- Estimation of network parameters for a given application based on analytical results.
- Design of a motion detector which is capable of real time performance using the proposed temporal module, and evaluation of its performance.
- Introduction of a systematic and modular training algorithm for larger networks that use the proposed structure.
- Investigation of the performance of a network that uses either a serial or a hierarchical connection of temporal modules.
- Investigating the feasibility of implementing a temporal module on a VLSI chip.

## 8.3. Future Work

A temporal module is trained to recognize a sequence of events by adjusting the interconnect weights in small increments at each training iteration to minimize the error between the actual output and the desired output. Implementation of such an interconnect requires a considerable number of transistors. Due to the limited number of transistors that can be fabricated on a chip, each interconnect weight is likely to be restricted to a resolution of only a few bits. As a result, future work is needed to evaluate the learning capability and the performance of a network using low precision weights.

The architecture of the current motion detector provides a discrete time differentiator for each pixel in the image input to detect a moving edge. An output of one is generated by a differentiator if the present input to a pixel is different from its previous value. Hence, the detector is not only sensitive to a moving edge, but also to changes occurring in the background. It works well only when there is very little relative movement between the imaging device (the observer) and the background. Further work is required to modify the architecture of the motion detector so that the detector can tolerate slow changes in the background.

There is an interesting application of the hierarchical network described in this thesis in speech recognition. The modules in the first level of the hierarchy can be used for recognition of different words and those in the second level for recognition of different sentences. More work is required to study the feasibility of this application.

# Bibliography


(1) Abu-Mostafa Y.S., 1989. Information theory, complexity and neural networks, *IEEE Communication Magazine*, Nov. pp. 25-28.

(2) Ackley D.H., Hinton G.E. and Sejnowski T.J. 1985. A learning algorithm for the Boltzmann machine, *Cognitive Science* Vol. 9, pp. 147-169.

(3) Adelson E.H. and Bergen J.R. 1986. The extraction of spatio-temporal energy in human and machine vision, *IEEE Proc. Workshop on Motion: Representation and Analysis*, May, pp. 151-155.

(4) Aggarwal J.K. 1986. Motion and time-varying imagery - an overview, *IEEE Proc. Workshop on Motion: Representation and Analysis*, May, pp. 1-5.

(5) Alspector J. 1989. Neural-style microsystems that learn, *IEEE Communication Magazine*, Nov. pp. 29-36.

(6) Ballard D.H., Hinton G.E. and Sejnowski T.J. 1984. Parallel visual computation, *Nature*, Vol. 306, pp. 21-36.

(7) Barlow H.B., Hill R.M. and Levick W.R. 1964. Retinal ganglion cells responding selectively to direction and speed of image motion in the rabbit. *Jour. Physiology*, Vol. 173, pp. 377-407.

(8) Barlow H.B. and Levick W.R. 1965. The mechanism of directionally selective units in rabbit's retina, *Jour. Physiology*, Vol. 178, pp. 477-504.

(9) Bengio Y., Cardin R., de Mori R. and Merlo E. 1989. Programmable execution of multi-layered networks for automatic speech recognition, *Communications of the ACM*, Vol. 32, pp. 195-199.

(10) Boahen K.A., Pouliquen P.O, Andreou A.G. and Jenkins J.L. 1989. A heteroassociative memory using current-mode MOS analog VLSI circuits, *IEEE Transaction on Circuits and Systems*, Vol. 36-5, pp. 747-761.

(11) Bottou L., Fogelman S.L., Blanchet P. and Lienard J.S. 1990. Speaker-independent isolated digit recognition: multilayer perceptrons vs. dynamic time warping, *Neural networks*, Vol. 3, pp. 453-465.

(12) Brown T.X. 1989. Neural networks for switching, *IEEE Communication Magazine*, Nov. pp. 72-81.

(13) Burt P.J., Bergen J.R., Hingorani R., Kolczynski R., Lee W.A., Leung A., Lubin J. and Shvaytser H. 1989. Object tracking with a moving camera: An application of dynamic motion analysis, *IEEE Proc. Workshop on Visual Motion* pp. 2-12.

(14) Curlander J.C. and Marmarelis V.Z. 1987. A linear spatio-temporal model of the light-to-bipolar cell system and its response characteristics to moving bars, *Biological Cybernetics*, Vol. 57, pp. 357-363.

(15) Dickmanns E.D. and Graefe V. 1988a. Dynamic monocular machine vision, *Machine Vision and Applications*, Vol. 1, pp. 223-240.

(16) Dickmanns E.D. and Graefe V. 1988b. Applications of dynamic monocular machine vision, *Machine Vision and Applications*, Vol. 1, pp. 241-261.

(17) Dickmanns E.D. 1989. Subject-Object Discrimination in 4D-Dynamic Scene Interpretation for Machine Vision, *IEEE Proc. Workshop on Visual Motion* pp. 298-304.

(18) Dowling J.E. 1987. *The retina - an approachable part of the brain*, The Belknap Press of Harvard University Press, Cambridge, Massachusetts, and London, England.

(19) Egelhaal M., Hausen K., Reichart W. and Wehrhahn C. 1988. Visual course control in flies relies on the neuronal computational of objects and background motion, *TINS*, Vol. 11-8, pp. 351-358.

(20) Feldman J.A. and Ballard D.H. 1982. Connectionist models and their properties, *Cognitive Science*, Vol. 6, pp. 205-254.

(21) Feldman J.A. 1985. Connectionist models and their applications: introduction, *Cognitive Science*, Vol. 9, pp. 1-2.

(22) Feldman J.A., Fanty M.A. and Goddard N.H. 1988. Computing with structured neural networks, *IEEE Computer*, Vol. 21, pp. 91-102.

(23) Felten E.W., Martin O., Otto S.W. and Hutchinson J. 1990. Multi-scale training of a large backpropagation net, *Biological Cybernetics*, Vol. 62, pp. 503-509.

(24) Fukushima K. 1975. A self-organizing multilayered neural network, *Biological Cybernetics*, Vol. 20, pp. 121-136.

(25) Fukushima K. 1980. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biological Cybernetics*, Vol. 36, pp. 193-202.

(26) Fukushima K. 1988. A neural network for visual pattern recognition, *IEEE Computer*, Vol. 21, pp. 65-75.

(27) Fukushima K. 1988. "Neocognitron: a hierarchical neural network capable of visual pattern recognition, *Neural Network*, Vol. 1, pp. 119-130.

(28) Girosi F., Verri A. and Torre V. 1989. Constraints for the computation of optical flow, *IEEE Proc. Workshop on Visual Motion*, pp. 116-124.

(29) Graf H.P., Lawrence D.J. and Hubbard W.E. 1988. VLSI implementation of a neural network model *IEEE Computer*, Vol. 21, pp. 41-49.

(30) Graf H.P. and Jackel L.D. 1989. Analog electronic neural network circuits, *IEEE Circuits and Devices Magazine*, Vol. 5-4, pp. 44-49.

(31) Grossberg, S. 1969. Some networks that can learn, remember and reproduce any number of complicated space-time patterns, I, *Journal of Mathematics and Mechanics*, Vol. 19, pp. 53-91.

(32) Grossberg, S. 1970. Some networks that can learn, remember and reproduce any number of complicated space-time patterns, II", *Studies in Applied Mathematics*, Vol. 49, pp. 135-136.

(33) Kohonen T. 1984. *Self-Organization and Associative Memory*, Springer-Verlag Press, Berlin.

(34) Habib M.K. and Akel H. 1989. A digital neuron-type processor and its VLSI design, *IEEE Transaction on Circuits and Systems*, Vol. 36-5, pp. 739-746.

(35) Hampshire J.B. and Waibel A.H. 1990. A novel objective function for improved phoneme recognition using time-delay neural networks, *IEEE Trans. Neural Network*, Vol. 1, pp. 216-228.

(36) Hinton G.E., Sejnowski T.J. and Ackley D.H. 1984. Boltzmann machines: constraint satisfaction networks that learn, *Technical Report CMU-CS-84-119*, Carnegie-Mellon University, Pittsburg PA.

(37) Hinton G.E. 1987. Connectionist learning procedures, *Technical Report CMU-CS-87-115*, Carnegie-Mellon University, Pittsburg PA.

(38) Hinton G.E. 1989. "Deterministic Boltzmann learning performs steepest descent in weight space, *Neural Computation*, Vol. 1, pp. 143-150.

(39) Hopfield J.J. 1982. Neural networks and physical systems with emergent collective computational abilities, *Proc. National Academy of Sciences USA*, Vol. 79, pp. 2554-2558.

(40) Hopfield J.J. and Tank D.W. 1986. Computing with neural circuits: a model, *Science*, Vol. 233, pp. 625-633.

(41) Hopfield J.J. 1988. Artificial neural network, *IEEE Circuits and Devices Magazine*, Sept.

(42) Hutchinson J., Koch C., Luo J. and Mead C. 1988. Computing motion using analog and binary resistive networks, *IEEE Computer*, Vol. 21, pp. 52-63.

(43) Jennings P.J. and Keele S.W. 1989. A computational model of attentional requirements in sequence learning, *Proc. Intl. Conf. Cognitive Science Society*, pp. 876-883.

(44) Jordan M.I. 1986. Serial order: A parallel distributed processing approach, *Technical report 8604*, Institute for Cognitive Science, University of California, San Diego.

(45) Kandel E.R. and Schwartz J.H. 1985. *Principles of neural science*, Elsevier, New York.

(46) Kirpatrick S., Gelatt Jr. C.D., Jr., Vecchi M.P. 1983. Simulated annealing, *Science* Vol. 220, pp. 671-674.

(47) Koch C., Wang H.T., Mathur B., Hsu A. and Suarez H. 1989. Computing optical flow in resistive networks and in the primate visual system, *IEEE Proc. Workshop on Visual Motion*, pp. 62-72.

(48) Kwan H.C., Yeap T.H., Jiang B.C. and Borrett D. 1990. Neural network control of simple limb movements, *Cdn. Jour. Physiol. Pharmacol.*, Vol. 68, pp. 126-130.

(49) Kung S.Y. 1988. Parallel architecture for artificial neural nets, *IEEE Proc. Intl. Conf. Neural Networks*, pp. II-165 - II-172.

(50) Lang K. and Hinton G.E. 1988. A time-delay neural network architecture for speech recognition, *Techical Report CMU-CS-88-152*, Carnegie-Mellon University, Pittsburgh PA.

(51) Lappin J.S. and Bell H.H. 1976. The detection of coherence in moving random-dot patterns, *Vision Res.*, Vol. 16, pp. 161-168.

(52) Lazarro J. and Mead C.A. 1989. A silicon model of auditory localization, *Neural Computation*, Vol. 1, pp. 47-57.

(53) LeCun Y., Boser B., Denker J.S., Henderson D., Howard R.E., Hubbard W. and Jackel L.D. 1989. Handwritten digit recognition with a back-propagation network, *IEEE Proc. Conf. Neural Information Processing Systems*, Nov.

(54) LeCun Y., Jackel L.D., Boser B., Denker J.S., Graf H.P. and Guyon I. 1989. Handwritten digit recognition: applications of neural network chips and automatic learning, *IEEE Communication Magazine*, Nov. pp. 41-46.

(55) Lee D., Papageorgiou A. and Wasilkowski G.W. 1989. Computing optical flow, *IEEE Proc. Workshop on Visual Motion*, pp. 99-106.

(56) Lippmann R. 1989. Pattern classification using neural networks, *IEEE Communication Magazine*, Nov. pp. 47-64.

(57) Lippmann R. 1989. Review of neural networks for speech recognition, *Neural Computation*, Vol. 1, pp. 1-38.

(58) Longuet-Higgins H.C. and Prazdny K. 1980. The interpretation of a moving retinal image, *Proc. Roy. Soc. Lond.*, B208, pp. 385-387.

(59) Maher M.A., DeWeerth S., Mohawald M.A. and Mead C.A. 1989. Implementing neural architectures using analog VLSI circuits, *IEEE Transaction on Circuits and Systems*, Vol.

36-5, pp. 643-652.

(60) Marshall J.A., 1990. Self-organizing neural networks for perception of visual motion, *Neural Networks*, Vol. 3, pp. 45-74.

(61) Massone L. and Bizzi E. 1989. A neural network model for limb trajectory formation, *Biological Cybernetics*, Vol. 61, pp. 417-425.

(62) McCulloch W.S. and Pitts W. 1943. A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133.

(63) Mead C. and Conway L. 1980. *Introduction to VLSI systems*, Addison-Wesley Publishing Company, Reading MA.

(64) Mead C.A. and Mahowald M.A. 1988. A silicon model of early visual processing, *Neural Networks*, Vol. 1-1, pp. 91-98.

(65) Mead C. 1989. *Analog VLSI and Neural Systems*, Addison-Wesley Publishing Company, Reading MA.

(66) Minsky M. and Papert S. 1969. *Perceptrons*, MIT press, Cambridge MA.

(67) Mozer M.C. 1988. A focused back-propagation algorithm for temporal pattern recognition, *Technical Report CRG-TR-88-3*, University of Toronto, Canada.

(68) Ogmen, H., Gagne, S., 1990. Neural network architectures for motion perception and elementary motion detection in the fly visual system, *Neural Networks*, Vol. 3, pp. 487-505.

(69) Pearlmutter B.A. 1989. Learning state space trajectories in recurrent neural networks, *IEEE Intl. Conf. Neural Networks* , Vol. 2, pp. 365-372.

(70) Poggio T. and Koch C. 1987. Synapses that compute motion, *Scientific American*, May, pp. 46-52.

(71) Psaltis D., Yamamura A., Hsu K., Lin S., Gu X.G. and Brady D. 1989. Optoelectronic implementations of neural networks, *IEEE Communication Magazine*, Nov. pp. 37-40.

(72) Reed R.D. and Geiger R.L. 1989. A multiple-input OTA circuit for neural networks, *IEEE Transaction on Circuits and Systems*, Vol. 36-5, pp. 767-770.

(73) Regan D. and Beverley K.I. 1984. Figure-ground segregation by motion contrast and by luminance contrast, *Optical Society of America A.*, Vol. 1, pp. 433-441.

(74) Roitblat H.L., Moore P.W.B., Nachtigall P.E., Penner R.H. and Au W.W.L. 1989. Dolphin echlocation: identification of returning echoes using a counterpropagation network, *IEEE Proc. Intl. Conf. Neural Networks*, pp. I-295 - I-299.

(75) Rumelhart D.E. and McClelland D.E. (eds.) 1986. *Parallel Distributed Processing*, MIT Press, Cambridge MA.

(76) Rumelhart D.E., Hinton G.E. and Williams R.J. 1986a. Learning internal representation by error propagation. In Rumelhart D.E. and McClelland D.E. (eds.), *Parallel Distributed Processing*, MIT Press, Cambridge MA., pp. 318-362.

(77) Rumelhart D.E., Hinton G.E. and Williams R.J. 1986b. Learning representation by back-propagation errors, *Nature*, Vol. 323-9, pp. 533-536.

(78) Schunck B.G. 1984a. The motion constraint equation for optical flow, *Proc. Intl. J. Conf. Pattern Recog.*, Montreal, Canada, pp. 20-22.

(79) Schunck B.G. 1984b. Motion segmentation and estimation by constraint line clustering, *Proc. Wkshp. on Compter Vision*, Annapolis, Maryland, pp. 58-62.

(80) Sejnowski T.J. and Rosenberg C.R. 1987. Parallel networks that learn to pronounce English text, *Complex Systems*, Vol. 1, pp. 145-168.

(81) Sereno M.I., 1989. Learning the solution to the aperture problem for pattern motion with a Hebb rule. In Touretzky D.S. (ed.), *Advances in neural information processing system I.*

(82) Shamma S. 1989. Spatial and temporal processing in central auditory networks. In Koch C. and Segev I. (eds.), *Methods in Neuronal Modelling*, MIT press/Bradford book, Cambridge, MA., pp. 247-289.

(83) Simon H.A. 1962. The architecure of complexity, *Proc. American Philosophical Society*, Vol. 106-6, pp. 467-482.

(84) Sivilotti M.R., Emerling C.A. and Mead C.A. 1986. VLSI architecture for implementation of neural networks, *American Institute of Physics Conference Proc.*, Vol. 151, pp. 408-413.

(85) Snyder M.A. 1989. On the mathematical foundations of smoothness constraints for the determination of optical flow and for surface reconstruction, *IEEE Proc. Workshop on Visual Motion*, pp. 107-115.

(86) Suzuki Y. and Atlas L.E. 1989. A study of regular architecture for digital implementation of neural networks, *IEEE Proc. Intl. Symp. Circuits and Systems*, pp. 82-85.

(87) Tank D.W. and Hopfield J.J. 1987. Neural computation by concentrating information in time, *Proc. Natural Academy of Science, USA*, Vol. 84, pp. 1896-1900.

(88) Tank D.W. and Hopfield J.J. 1986. Simple neural optimization networks: a A/D converter, signal decision circuit, and a linear programming circuit, *IEEE Trans. on Circuits and Systems*, Vol. CAS-33-5, pp. 533-541.

(89) Tomberg J., Ritoniemi T., Tenhunen H. and Kaski K. 1989. VLSI implementation of pulse density modulated neural network structure, *IEEE. Proc. Intl. Symp. on Circuits and Systems*, pp. 2104-2107.

(90) Tou J.T. and Gonzalez R.C. 1974. *Pattern Recognition Principles*, Addison-Wesley Publishing Company, Reading MA.

(91) Tsotsos J.K., Mylopoulous J.C. and Zucker S. 1980. A framework for visual motion understanding, *IEEE Pattern Analysis and Machine Intelligence*, Vol. 2, pp. 563-573.

(92) Tsotsos J.K. 1981. Temporal event recognition: an application to left ventricular performance assessment, *Proc. ICJAI*, Vancouver.

(93) Tsotsos J.K. 1987. Representational axes and temporal cooperative processes, *Vision, Brain and Cooperative Computation*, ed. Arbib M.A., and Allen R., MIT press/Bradford Books, Cambridge MA. pp. 361-418.

(94) Tsotsos J.K. 1988. A complexity level analysis of immediate vision, *Intl. Jour. Computer Vision*, Vol. 1-4, pp. 303-320.

(95) Tsotsos J.K. 1989. The complexity of perceptual search task, *Technical Report RBCV-TR-89-28*, University of Toronto, Canada.

(96) Van Den Bout D.E. and Miller T.K. 1989. A digital architecture employing stochasticism for the simulation of Hopfiled neural nets, *IEEE Trans. on Circuits and Systems*, Vol. 36-5, pp. 732-738.

(97) Verleysen M., Sirletti B., Vandemeulebroecke A. and Jespers P.G.A. 1989. A high-storage capacity content addressable memory and its learning algorithm, *IEEE Trans. on Circuits and Systems*, Vol. 36-5, pp. 762-766.

(98) Vladimirescu A., Zhang K., Newton A.R., Pederson D.O. and Sangiovanni-Vincentell, A. 1981. *SPICE version 2G user manual*, University of California, Berkeley CA.

(99) Waibel A. 1988. Modular construction of time-delay neural networks for speech recognition, *Neural Computation*, Vol. 1, pp. 39-46.

(100) Wang H.T., Mathur B. and Koch C. 1989. Computing Optical Flow in the Primate Visual System, *Neural Computation*, Vol. 1, pp. 92-103.

(101) Watrous R.L. and Lokendra S. 1987. Learning phoenetic features using connectionist networks: an experiment in speech recognition, *IEEE Proc. Intl. Conf. on Neural Networks*, pp. IV-381.

(102) Waxman A.M., Wu J., and Seibert M. 1989. Computing visual motion in the short and the long: From receptive field to neural networks, *IEEE Proc. Workshop on Visual Motion*, pp. 156-162.

(103) Widrow B. and Stearns S.D. 1985. *Adaptive Signal Processing*, Prentice-Hall, Englewood-Cliff, N.J.

(104) Williams R.J. and Zipser D. 1989. A learning algorithm for continually running fully recurrent neural networks, *Neural Computation*, Vol. 1, pp. 270-280.

(105) Widrow B. and Winter R. 1988. Neural nets for adaptive filtering and adaptive pattern recognition, *IEEE Computer*, Vol. 21, pp. 25-39.

(106) Wolfram S. 1986. Approaches to complexity engineering, *Physica*, Vol. 22D, pp. 385-399.

(107) Yeap T.H., Zaky S.F., Tsotsos J.K. and Kwan H.C. 1989. A neural network for temporal pattern recognition, *IEEE Proc. Intl. Symp. Circuits and Systems*, pp. 778-781.

(108) Yeap T.H., Zaky S.F., Tsotsos J.K. and Kwan H.C. 1990. A hierarchical neural network for temporal pattern recognition, to appear in *IEEE Proc. Intl. Symp. Circuits and Systems*, pp. 2967-2970.

(109) Yeap T.H., Tsotsos J.K., Zaky S.F. and Kwan H.C. 1990. A motion detector based on a neural network, to appear in *Proc. Canadian Conf. on Electrical and Computer Engineering*.

(110) Yuhas B.P., Goldstein M.H. and Sejnowski T.J. 1989. Integration of acoustic and visual speech signals using neural networks, *IEEE Communication Magazine*, Nov. pp. 65-71.

(111) Yuille A.L. and Grzywacz N.M. 1988. A computational theory for the perception of coherent visual motion, *Nature*, Vol. 333, pp. 71-74.

(112) Zitti E.D., Bisio G.M. and Cavaglia D.D. 1989. Analysis of neural algorithms for parallel architectures, *IEEE Proc. Intl. Symp. on Circuits and Systems*, pp. 2197-2200.

# Appendix A

# Weight Adjustment Function

This appendix presents the derivation of the function used to adjust the interconnect weights of a node. The function is used in the training algorithm presented in chapter 3.

## A.1. Constant Inputs

A node may be connected to a number of inputs by weighted connections, as shown in Fig. A.1 . When subjected to an input pattern $p$, the internal state $x_{pi}$ of node $i$ changes with a fixed time constant $\tau$ and can be expressed in the following as

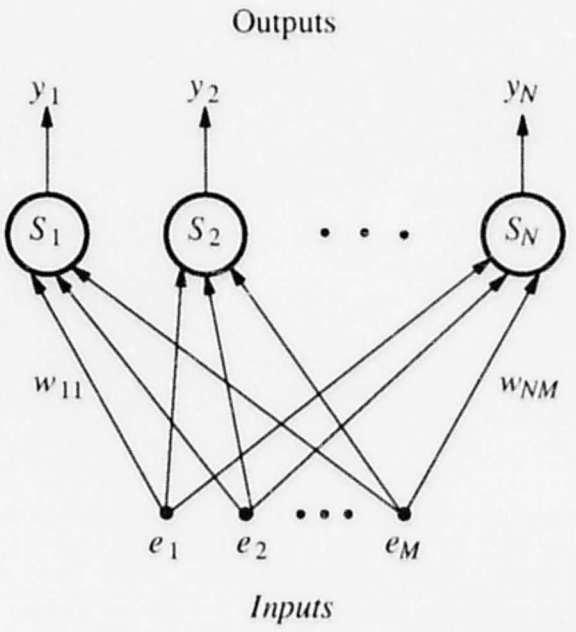



Fig. A.1 Nodes linked to inputs by weighted interconnections.

$$\tau \frac{dx_{pi}}{dt} + x_{pi} = \sum_{j=1}^{N} w_{ij} e_j + \theta_i \tag{A.1}$$

where $\tau$ is the time constant of a node, $w_{ij}$ is the weight of the connection from input $e_j$ to the input of node $i$ and $\theta_i$ is the internal bias of node $i$. Node $i$ has a sigmoidal output $y_{pi}$ in response to its internal state, as defined in the following.

$$y_{pi} = \frac{1}{1 + e^{-x_{pi}}} \tag{A.2}$$

Let $E$ be the total sum of the square of errors of all nodes for $M$ input patterns given by

$$E = \frac{1}{2} \sum_{p=1}^{M} \sum_{i=1}^{N} (\hat{y}_{pi} - y_{pi})^2 \tag{A.3}$$

where $\hat{y}_{pi}$ and $y_{pi}$ are the $i$th component of the desired output and actual output vectors of the network respectively, and $N$ is the number of nodes in the network. The objective is to minimize $E$ with respect to $w_{ij}$. From Eqns. A.2 and A.3, we have

$$\frac{\delta E}{\delta y_{pi}} = \hat{y}_{pi} - y_{pi}$$

and

$$\frac{\delta y_{pi}}{\delta x_{pi}} = y_{pi}(1 - y_{pi})$$

Therefore,

$$\frac{\delta E}{\delta x_{pi}} = y_{pi}(\hat{y}_{pi} - y_{pi})(1 - y_{pi}) \tag{A.4}$$

Next, we need to determine $\frac{\delta x_{pi}}{\delta w_{ij}}$. In order to do so, we assume that input $e_j$ is constant over a sampling interval $T$. Solving Eqn. A.1 using the Laplace tranformation, we obtain,

$$x_{pi} = (\sum_{j=1}^{N} w_{ij} e_j + \theta_i)(1 - e^{\frac{-t}{\tau}}) \tag{A.5}$$

Differentiating Eqn. A.5 with respect to $w_{ij}$, we get

$$\frac{\delta x_{pi}}{\delta w_{ij}} = e_j(1 - e^{\frac{-t}{\tau}}) \tag{A.6}$$

Substituting Eqns. A.4 and A.6 into Eqn. A.4, we obtain

$$\frac{\delta E}{\delta w_{ij}} = e_j y_{pi}(\hat{y}_{pi} - y_{pi})(1 - y_{pi})(1 - e^{\frac{-t}{\tau}}) \tag{A.7}$$

If all the weights are adjusted at the end of each sampling interval $T$, then $\frac{\delta E}{\delta w_{ij}}$ can be approximated by the following equation

$$\begin{aligned} \frac{\delta E}{\delta w_{ij}} &= e_j y_{pi}(\hat{y}_{pi} - y_{pi})(1 - y_{pi})(1 - e^{\frac{-T}{\tau}}) \\ &\approx \eta e_j y_{pi}(\hat{y}_{pi} - y_{pi})(1 - y_{pi}) \end{aligned} \tag{A.8}$$

where $\eta$ is given by

$$\eta = H(1 - e^{\frac{-T}{\tau}})$$

and $H$ is the training step size with the value between 0 and 1.

Eqn. A.8 is used to adjust the input interconnect weights linking the external inputs and the sequence node in a module, and the bias in each node.

## A.2. Exponentially Rising Inputs

Next, we want to find out if Eqn. A.8 can be applied to cases when input $e_j$ is not constant. There are two interesting cases: when the input is exponetially rising from 0 towards 1 or decaying from 1 towards 0 by a time constant $\tau$. These cases correspond to the situation when the input of a node is connected to the output of another node. We shall examine in this section the case when input $e_j$ is an exponentially rising variable given by

$$e_j(t) = I_j(1 - e^{\frac{-t}{\tau}}) \tag{A.9}$$

where $I_j$ is a constant between 0 and 1. Substituting Eqn. A.9 into Eqn. A.1 and solving using the

Laplace transformation, the following expression is obtained.

$$x_{pi} = (\sum_{j=1}^{N} w_{ij} I_j + \theta_i)(1 - e^{\frac{-t}{\tau}}) - \sum_{j=1}^{N} w_{ij} I_j (\frac{t}{\tau}) e^{\frac{-t}{\tau}} \tag{A.10}$$

Differentiating Eqn. A.10 with respect to $w_{ij}$ yields

$$\frac{\delta x_{pi}}{\delta w_{ij}} = I_j (1 - e^{\frac{-t}{\tau}} - \frac{t}{\tau} e^{\frac{-t}{\tau}}) \tag{A.11}$$

Similarly, if all the weights are adjusted at the end of each sampling interval $T$, then by substituting $t = T$, Eqns. A.6 and A.13 into Eqn. A.4, we obtain

$$\begin{aligned} \frac{\delta E}{\delta w_{ij}} &= I_j y_{pi} (\hat{y}_{pi} - y_{pi})(1 - y_{pi})(1 - e^{\frac{-T}{\tau}} - \frac{T}{\tau} e^{\frac{-T}{\tau}}) \\ &\approx \eta I_j y_{pi} (\hat{y}_{pi} - y_{pi})(1 - y_{pi}) \end{aligned} \tag{A.12}$$

where $\eta$ is given by

$$\eta = H(1 - e^{\frac{-T}{\tau}} - \frac{T}{\tau} e^{\frac{-T}{\tau}})$$

and $H$ is the training step size with a value between 0 and 1.

This function can be used to adjust the weight of the excitatory connection from sequence node $i$ to $i+1$.

## A.3. Exponentially Decaying Inputs

Now, we examine the second case when input $e_j$ is an exponentially decaying output defined by the following.

$$e_j(t) = I_j e^{\frac{-t}{\tau}} \tag{A.13}$$

Substituting Eqn. A.13 into Eqn. A.1 and solving using the Laplace transformation gives the following expression

$$x_{pi} = \theta_i(1 - e^{\frac{-t}{\tau}}) - \sum_{j=1}^{N} w_{ij} I_j (\frac{t}{\tau}) e^{\frac{-t}{\tau}} \tag{A.14}$$

Differentiating Eqn. A.16 with respect to $w_{ij}$ yields

$$\frac{\delta x_{pi}}{\delta w_{ij}} = I_j \frac{t}{\tau} e^{\frac{-t}{\tau}} \tag{A.15}$$

Substituting $t = T$, Eqns. A.6 and A.17 into Eqn. A.4 gives

$$\frac{\delta E}{\delta w_{ij}} = I_j y_{pi} \frac{T}{\tau} e^{\frac{-T}{\tau}} (\hat{y}_{pi} - y_{pi})(1 - y_{pi}) \tag{A.16}$$

$$\approx \eta I_j y_{pi} (\hat{y}_{pi} - y_{pi})(1 - y_{pi})$$

where $\eta$ is given by

$$\eta = H(\frac{T}{\tau}) e^{\frac{-T}{\tau}}$$

and $H$ is the training step size with a value between 0 and 1.

This function is used to adjust the weights of all inhibitory weights between the sequence nodes.

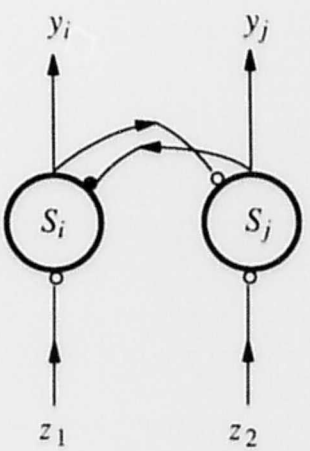
# Appendix B

## Analysis of Sequence Nodes

The solutions to the linear analysis of sequence nodes presented in Section 4.3.1. of chapter 4 is given in this appendix.

The dynamic equations for nodes $S_i$ and $S_j$ can be expressed in the form

$$\tau_s \frac{dx_i}{dt} + x_i = bz_i - ay_j + \theta_s \qquad \text{(B.1)}$$




○ excitatory connection
● inhibitory connection

Fig. B.1 Two sequence nodes.

$$\tau_s \frac{dx_j}{dt} + x_j = bz_j + ay_i + \theta_s \tag{B.2}$$

$$y_j = \frac{x_j}{5} + \frac{1}{2} \tag{B.3}$$

where $\tau_s$ is the time constant associated with each sequence node, $b$ is the weight of the external input link, $a$ is the weight of the link connecting the output of one node to the input of another node, $z_k$, is the external input $k$, and $\theta_s$ is the bias.

Substituting Eqn. B.3 into Eqns. B.1 and B.2 yields the following.

$$\tau_s \frac{dx_i}{dt} + x_i = bz_i - \frac{ax_j}{5} + (\theta_s - \frac{a}{2}) \tag{B.4}$$

$$\tau_s \frac{dx_j}{dt} + x_j = bz_j + \frac{ax_i}{5} + (\theta_s + \frac{a}{2}) \tag{B.5}$$

By applying the Laplace transformation to Eqns. B.4 and B.5, we obtain

$$X_i(s) = \frac{bZ_i(s) - \frac{aX_j(s)}{5}}{\tau_s(s+\frac{1}{\tau_s})} + \frac{(\theta_s - \frac{a}{2})}{\tau_s s(s+\frac{1}{\tau_s})} + \frac{x_i(0)}{(s+\frac{1}{\tau_s})} \tag{B.6}$$

$$X_j(s) = \frac{bZ_j(s) + \frac{aX_i(s)}{5}}{\tau_s(s+\frac{1}{\tau_s})} + \frac{(\theta_s + \frac{a}{2})}{\tau_s s(s+\frac{1}{\tau_s})} + \frac{x_j(0)}{(s+\frac{1}{\tau_s})} \tag{B.7}$$

where $x_i(0)$ and $x_j(0)$ are the initial states of nodes $S_i$ and $S_j$, respectively. Solving Eqns. B.6 and B.7, the following expressions for $X_i(s)$ and $X_j(s)$ are obtained.

$$\begin{aligned} X_i(s) = {} & \frac{bZ_i(s)(s+\frac{1}{\tau_s})}{\tau_s((s+\frac{1}{\tau_s})^2+\omega^2)} + \frac{(\theta_s - \frac{a}{2})(s+\frac{1}{\tau_s})}{\tau_s s((s+\frac{1}{\tau_s})^2+\omega^2)} + \frac{x_i(0)(s+\frac{1}{\tau_s})}{(s+\frac{1}{\tau_s})^2+\omega^2} - \\ & \frac{abZ_j(s)}{5\tau_s^2((s+\frac{1}{\tau_s})^2+\omega^2)} - \frac{a(\theta_s+\frac{a}{2})}{5\tau_s^2 s((s+\frac{1}{\tau_s})^2+\omega^2)} - \frac{ax_j(0)}{5\tau_s((s+\frac{1}{\tau_s})^2+\omega^2)} \end{aligned} \tag{B.8}$$

$$X_j(s) = \frac{bZ_j(s)(s+\frac{1}{\tau_s})}{\tau_s((s+\frac{1}{\tau_s})^2+\omega^2)} + \frac{(\theta_s+\frac{a}{2})(s+\frac{1}{\tau_s})}{\tau_s s((s+\frac{1}{\tau_s})^2+\omega^2)} + \frac{x_j(0)(s+\frac{1}{\tau_s})}{(s+\frac{1}{\tau_s})^2+\omega^2} + \frac{abZ_i(s)}{5\tau_s^2((s+\frac{1}{\tau_s})^2+\omega^2)} + \frac{a(\theta_s-\frac{a}{2})}{5\tau_s^2 s((s+\frac{1}{\tau_s})^2+\omega^2)} + \frac{ax_i(0)}{5\tau_s((s+\frac{1}{\tau_s})^2+\omega^2)} \tag{B.9}$$

where

$$\omega = \frac{a}{\tau_s} \tag{B.10}$$

If the external inputs presented are

$$z_i = u(t-(i-1)T) - u(t-iT) \tag{B.11}$$

$$z_j = u(t-(j-1)T) - u(t-jT) \tag{B.12}$$

where $T$ is the sampling interval, then the Laplace transformation of Eqns. B.11 and B.12 are given in the following as

$$Z_i(s) = \frac{e^{-(i-1)T}}{s} - \frac{e^{-iT}}{s} \tag{B.13}$$

$$Z_j(s) = \frac{e^{-(j-1)T}}{s} - \frac{e^{-jT}}{s} \tag{B.14}$$

Substituting Eqns. B.13 and B.14 yields the following responses for nodes $S_i$ and $S_j$.

$$\begin{aligned} x_i(t) = {} & e^{\frac{-t}{\tau_s}}(x_i(0)\cos\omega t - \frac{a}{5}x_j(0)\sin\omega t)u(t) \\ & + \frac{((\theta_s-\frac{a}{2})\beta_s(t) - \frac{a}{5}(\frac{a}{2}+\theta_s)\alpha_s(t))u(t)}{1-a^2} \\ & + \frac{b(\beta_s(t-(i-1)T)u(t-(i-1)T) - \beta_s(t-iT)u(t-iT)}{1-a^2} \\ & - \frac{ab(\alpha_s(t-(j-1)T)u(t-(j-1)T) - \alpha_s(t-jT)u(t-jT))}{5(1-a^2)} \end{aligned} \tag{B.15}$$

$$
\begin{aligned}
x_j(t) &= e^{\frac{-t}{\tau_s}}\left(x_j(0)\cos\omega t + \frac{a}{5}x_i(0)\sin\omega t\right)u(t) \\
&+ \frac{\left(\left(\theta_s + \frac{a}{2}\right)\beta_s(t) + \frac{a}{5}\left(\frac{a}{2} - \theta_s\right)\alpha_{s(t)}\right)u(t)}{1-a^2} \\
&+ \frac{b\left(\beta_s(t-(j-1)T)u(t-(j-1)T) - \beta_s(t-jT)u(t-jT)\right)}{1-a^2} \\
&+ \frac{ab\left(\alpha_s(t-(i-1)T)u(t-(i-1)T) - \alpha_s(t-iT)u(t-iT)\right)}{5(1-a^2)}
\end{aligned}
\tag{B.16}
$$

where

$$
\beta_s(t) = 1 - e^{\frac{-t}{\tau_s}}\cos\omega t - \tau_s\omega e^{\frac{-t}{\tau_s}}\sin\omega t \tag{B.17a}
$$

$$
\alpha_s(t) = 1 - e^{\frac{-t}{\tau_s}}\cos\omega t - \frac{1}{\tau_s}\omega e^{\frac{-t}{\tau_s}}\sin\omega t \tag{B.17b}
$$

Eqns. B.15 and B.16 correspond to Eqns. 4.7 and 4.8 of chapter 4, respectively.

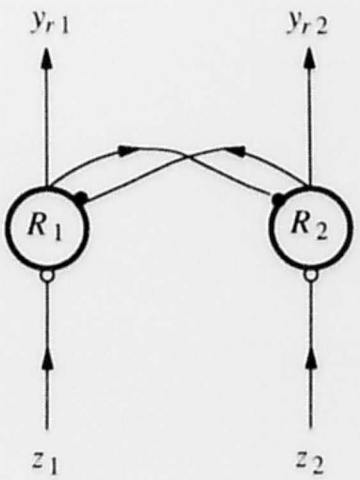
# Appendix C

## Analysis of Competition Between Result Nodes

The solutions to the linear analysis of the competition between two result nodes to become a winner presented in section 4.4.2. of chapter 4 is given in this appendix.

The dynamic equations for nodes $R_1$ and $R_2$ can be expressed in the form

$$\tau_r \frac{dx_{r1}}{dt} + x_{r1} = cz_1 - ay_{r2} + \theta_r \qquad \text{(C.1)}$$




○ excitatory connection

● inhibitory connection

Fig. C.1 Competition between two result nodes.

$$\tau_r \frac{dx_{r2}}{dt} + x_{r2} = cz_2 - ay_{r1} + \theta_r \tag{C.2}$$

$$y_{rj} = \frac{x_{rj}}{5} + \frac{1}{2} \tag{C.3}$$

where $\tau_r$ is the integrating time-constant, $x_{ri}$, $y_{ri}$ are the internal state and output of node $i$ respectively, $c$ is the weight of the external input link, $z_i$ is the external input to node $i$, $a$ is the weight of inhibitory connections between the nodes,

Substituting Eqn. C.3 into Eqns. C.1 and C.2 yields the following.

$$\tau_r \frac{dx_{r1}}{dt} + x_{r1} = cz_1 - \frac{ax_{r2}}{5} + (\theta_r - \frac{a}{2}) \tag{C.4}$$

$$\tau_r \frac{dx_{r2}}{dt} + x_{r2} = cz_2 - \frac{ax_{r1}}{5} + (\theta_r - \frac{a}{2}) \tag{C.5}$$

By applying the Laplace transformation to Eqns. C.4 and C.5, we obtain

$$X_{r1}(s) = \frac{cZ_1(s) - \frac{aX_{r2}(s)}{5}}{\tau_r(s+\frac{1}{\tau_r})} + \frac{(\theta_s - \frac{a}{2})}{\tau_r s(s+\frac{1}{\tau_r})} + \frac{x_{r1}(0)}{(s+\frac{1}{\tau_r})} \tag{C.6}$$

$$X_{r2}(s) = \frac{cZ_2(s) - \frac{aX_{r1}(s)}{5}}{\tau_r(s+\frac{1}{\tau_r})} + \frac{(\theta_s - \frac{a}{2})}{\tau_r s(s+\frac{1}{\tau_r})} + \frac{x_{r2}(0)}{(s+\frac{1}{\tau_r})} \tag{C.7}$$

where $x_{r1}(0)$ and $x_{r2}(0)$ are the initial states of nodes $R_1$ and $R_2$, respectively. Solving Eqns. C.6 and C.7, the following expressions for $X_{r1}(s)$ and $X_{r2}(s)$ are obtained.

$$\begin{aligned} X_{r1}(s) = {} & \frac{cZ_1(s)(s+\frac{1}{\tau_r})}{\tau_r((s+\frac{1}{\tau_r})^2-\omega^2)} + \frac{(\theta_s - \frac{a}{2})(s+\frac{1}{\tau_r})}{\tau_r s((s+\frac{1}{\tau_r})^2-\omega^2)} + \frac{x_{r1}(0)(s+\frac{1}{\tau_r})}{(s+\frac{1}{\tau_r})^2-\omega^2} - \\ & \frac{caZ_2(s)}{5\tau_r^2((s+\frac{1}{\tau_r})^2-\omega^2)} - \frac{a(\theta_s - \frac{a}{2})}{5\tau_r^2 s((s+\frac{1}{\tau_r})^2-\omega^2)} - \frac{ax_{r2}(0)}{5\tau_r((s+\frac{1}{\tau_r})^2-\omega^2)} \end{aligned} \tag{C.8}$$

$$X_{r2}(s) = \frac{cZ_2(s)(s+\frac{1}{\tau_r})}{\tau_r((s+\frac{1}{\tau_r})^2-\omega^2)} + \frac{(\theta_s-\frac{a}{2})(s+\frac{1}{\tau_r})}{\tau_r s((s+\frac{1}{\tau_r})^2-\omega^2)} + \frac{x_{r2}(0)(s+\frac{1}{\tau_r})}{(s+\frac{1}{\tau_r})^2-\omega^2} - \frac{caZ_1(s)}{5\tau_r^2((s+\frac{1}{\tau_r})^2-\omega^2)} - \frac{a(\theta_s-\frac{a}{2})}{5\tau_r^2 s((s+\frac{1}{\tau_r})^2-\omega^2)} - \frac{ax_{r1}(0)}{5\tau_r((s+\frac{1}{\tau_r})^2-\omega^2)} \quad (C.9)$$

where

$$\omega = \frac{a}{\tau_r} \quad (C.10)$$

If the external inputs presented are

$$z_1 = K_1 u(t) \quad (C.11)$$

$$z_2 = K_2 u(t) \quad (C.12)$$

where $K_1$ and $K_2$ are constants with the values between 0 and 1, then the Laplace transformation of Eqns. C.11 and C.12 are given in the following as

$$Z_1(s) = \frac{K_1}{s} \quad (C.13)$$

$$Z_2(s) = \frac{K_2}{s} \quad (C.14)$$

Substituting Eqns. C.13 and C.14 yields the following responses for nodes $R_1$ and $R_2$.

$$x_{r1}(t) = \frac{[(cK_1+\theta_r-\frac{a}{2})\beta_r(t)-\frac{a}{5}(cK_2+\theta_r-\frac{a}{2})\alpha_r(t)]u(t)}{1-w_m^2} + e^{\frac{-t}{\tau_r}}(x_{r1}(0)\cosh\omega t - \frac{a}{5}x_{r2}(0)\sinh\omega t)u(t) \quad (C.15)$$

$$x_{r2}(t) = \frac{[(cK_2+\theta_r-\frac{a}{2})\beta_r(t)-\frac{a}{5}(cK_1+\theta_r-\frac{a}{2})\alpha_r(t)]u(t)}{1-w_m^2} + e^{\frac{-t}{\tau_r}}(x_{r2}(0)\cosh\omega t - \frac{a}{5}x_{r1}(0)\sinh\omega t)u(t) \quad (C.16)$$

$$\beta_r(t) = 1 - e^{\frac{-t}{\tau_r}} \cosh \omega t - \tau_r \omega e^{\frac{-t}{\tau_r}} \sinh \omega t \tag{C.17a}$$

$$\alpha_r(t) = 1 - e^{\frac{-t}{\tau_r}} \cosh \omega t - \frac{1}{\tau_r \omega} e^{\frac{-t}{\tau_r}} \sinh \omega t \tag{C.17b}$$

Eqns. C.15 and C.16 correspond to Eqns. 4.24 and 4.25 of chapter 4, respectively.