A Modified ART 1 Algorithm more suitable for VLSI Implementations

Teresa Serrano-Gotarredona and Bernabé Linares-Barranco

National Microelectronics Center (CNM), Dept. of Analog Design, Ed. CICA, Av. Reina Mercedes s/n, 41012 Sevilla, SPAIN. Phone: 34-5-4239923, Fax: 34-5-4624506, E-mail: bernabe@cnm.us.es

Abstract

This paper presents a modification to the original ART 1 algorithm [Carpenter, 1987a] that is conceptually similar, can be implemented in hardware with less sophisticated building blocks, and maintains the computational capabilities of the originally proposed algorithm. This modified ART 1 algorithm (which we will call here ART 1_m) is the result of hardware motivated simplifications investigated during the design of an actual ART 1 chip [Serrano, 1994, 1996]. The purpose of this paper is simply to justify theoretically that the modified algorithm preserves the computational properties of the original one and to study the difference in behavior between the two approaches.

I. Introduction

In 1987 Carpenter and Grossberg published the ART 1 algorithm in a brilliant and well-founded paper [Carpenter, 1987a], the first of a series of **A**daptive **R**esonance **T**heory (ART) architectures. ART 1 is an architecture capable of learning (in an unsupervised way) recognition codes in response to arbitrary orderings of arbitrarily many and complex binary input patterns. The ART 2 [Carpenter, 1987b] and Fuzzy-ART [Carpenter, 1991a] architectures do the same but for analog input patterns. ART 3 [Carpenter, 1990] introduces a search process for ART architectures that can robustly cope with sequences of asynchronous analog input patterns in real time. ARTMAP [Carpenter, 1991b] and Fuzzy-ARTMAP [Carpenter, 1992] can be taught to learn (in a supervised way) predetermined categories of binary and analog input patterns, respectively. This paper focuses only on the ART 1 architecture. This architecture has a collection of interesting computational properties:

- *Self-Scaling:* The self-scaling property discovers critical features in a context-sensitive way. For example, if two binary input patterns have M bits set to '1', and all except for m of them are at the same location, these two different input patterns can be classified into the same category if m/M is sufficiently small, or as two different categories if m/M is not so small.
- *Vigilance or Variable Coarseness:* There is a vigilance parameter ($0 < \rho \leq 1$) that adjusts the coarseness of the categories that will be formed. If the vigilance parameter is set close to '1', more attention will be dedicated to distinguishing very similar input patterns and classifying and learning them as belonging to different categories. However, if the vigilance parameter is close to '0', there must be a significant difference between two input patterns for the system to separate them into two different categories.

Submitted to *Neural Networks* on November 24, 1994. Accepted on November 13, 1995.

- *Subset and Superset Direct Access:* Suppose the system has learned two different categories such that one is represented by a binary pixel image that is a subset of the image representing the other. The first is a subset of the second, which is a superset of the first. Under these circumstances, the system can classify a new input pattern as belonging to either the subset or the superset category, depending on global similarity criteria. No restrictions on input orthogonality or linear predictability are needed.
- *Stable Category Learning:* In response to an arbitrary list of binary input patterns, all interconnection weights subject to learning approach limits after a finite number of learning trials. Learning is guaranteed to stabilize, and it does so for a small number of training patterns presentations.
- *Biasing the Network to form New Categories:* When a new pattern arrives, a competition starts between stored patterns to capture it. One of the competing categories is the *empty* or *uncommitted* category. There exists a parameter that can bias the tendency of the *uncommitted* category to initially capture a new pattern, before the *vigilance parameter* plays any role.

In the original ART 1 paper [Carpenter, 1987a], the architecture is mathematically described as sets of Short Term Memory (STM) and Long Term Memory (LTM) time domain nonlinear differential equations. The STM differential equations describe the evolution of and interactions between processing units or neurons of the system, while the LTM differential equations describe how the interconnection weights change in time as a function of the state of the system. The time constants associated with the LTM differential equations are much slower than those associated with the STM differential equations. A valid assumption, also presented by Carpenter and Grossberg [Carpenter, 1987a], is to make the STM differential equations settle instantaneously to their corresponding steady state and consider only the dynamics of the LTM differential equations. In this case, the STM differential equations must be substituted by nonlinear algebraic equations that describe the corresponding steady state of the system. Furthermore, Carpenter and Grossberg also introduced the fast learning mode of the ART 1 architecture, in which the LTM differential equations are also substituted by their corresponding steady-state nonlinear algebraic equations. Thus, the ART 1 architecture, originally modelled as a dynamically evolving collection of neurons and synapses governed by time-domain differential equations, can be behaviorally modelled as the sequential application of nonlinear algebraic equations: an input pattern is given, the corresponding STM steady state is computed through the STM algebraic equations, and the system weights are updated using the corresponding LTM algebraic equations.

At this point three different levels of ART 1 implementations (in either software or hardware) can be distinguished:

Type-1 *Full Model Implementation:* Both STM and LTM time-domain differential equations are realized. This implementation is the most expensive and requires a large amount of computational power.

Type-2 *STM Steady-State Implementation:* Only the LTM time-domain differential equations are implemented. The STM behavior is governed by nonlinear algebraic equations. This implementation requires less resources than the previous one. However, proper sequencing of STM events must be assured, which is architecturally implicit in the *Type-1* implementation.

Type-3 *Fast Learning Implementation:* This implementation is computationally the least expensive. In this case, STM and LTM events must be algorithmically sequenced.

Regarding hardware implementations of the ART 1 architecture, several attempts have been reported in the literature. Ho et al. proposed a circuit technique for a *Type-1* implementation [Ho, 1994]; Tsay and Newcomb proposed a CMOS circuit technique that would realize a partial *Type-2* implementation [Tsay, 1991]; Wunsch et al. [Wunsch, 1993] have built an optical-based *Type-3* implementation; elsewhere [Serrano, 1994, 1996] we present a CMOS VLSI *Type-3* circuit.

This paper presents a modification to the original ART 1 algorithm [Carpenter, 1987a] (which we will call from now on ART 1_m, as referring to "ART 1 - modified") that is conceptually similar, can be implemented in hardware with less sophisticated building blocks, and maintains the same computational capabilities as the originally proposed algorithm. This modification was motivated by a *Type-3* hardware implementation and was investigated during the design process of an actual ART 1 *Type-3* chip [Serrano, 1994, 1996]. However, such modifications can be extended to *Type-2* and *Type-1* implementation versions as well, as shown at the end of this paper.

The paper is organized as follows: Section II develops the ART 1_m architecture starting from the original ART 1 *Type-3* (or *Fast Learning*) description and driven by hardware implementation considerations. Section III shows that all computational properties present in the original ART 1 architecture are preserved in the modified version. Section IV studies the differences in behavior between the two descriptions and provides simulation results, and Section V indicates how to extend the ART 1_m *Type-3* description to *Type-2* and *Type-1* models.

II. From the Original ART 1 Algorithm to the Modified One

Let us start by describing the *Type-3* model of the original ART 1 architecture. The ART 1 topology is shown in Fig. 1 and consists of two layers: layer F_1 is the input layer and has M nodes (one for each binary "pixel" of the input pattern), and layer F_2 is the category layer and has N nodes. Let us call the nodes in layer F_1 x_i, and the nodes in layer F_2 y_j. In the original ART 1 paper specific notations were used to distinguish between *internal state, output,* and *node name* for F_1 and F_2 nodes. In this paper, since we are concerned exclusively with *Type-3* descriptions, we will use a single notation to refer to either *internal state, output,* and *node name* of F_1 nodes (x_i) and F_2 nodes (y_j). Each node in the F_2 layer represents a "cluster" or "category". In this layer, only one node will become active after presentation of an input pattern $\mathbf{I} \equiv (I_1, I_2, \ldots, I_M)$. The F_2 layer category that will become active is that which most closely represents the input pattern $\mathbf{I}$. If no preexisting category is satisfactory for a given input pattern, a new category will be formed. Each F_1 node x_i is connected to all F_2 nodes y_j through bottom-up connections of weight[1] z_{ij}^{bu}, so that the input received by each F_2 node y_j is given by

$$T_j = \sum_{i=1}^{M} z_{ij}^{bu} I_i \ . \tag{1}$$

Layer F_2 acts as a Winner-Take-All network[2] so that all nodes y_j remain inactive, except that which receives the largest bottom-up input T_j,

$$\begin{aligned} y_J &= 1 \quad \text{if} \quad T_J = max_j \{T_j\} \ , \\ y_j &= 0 \quad \text{otherwise} \ . \end{aligned} \tag{2}$$

Once an F_2 winning node arises, a top-down pattern is activated through the top-down weights[3] z_{ji}^{td}. Let us call this top-down pattern $\mathbf{X} \equiv (X_1, X_2, \ldots, X_M)$. The resulting vector $\mathbf{X}$ is given by the equation,

$$X_i = I_i \sum_j z_{ji}^{td} y_j \ . \tag{3}$$

Since only one y_j is active, let us call this winning F_2 node y_J, so that y_j=0 if $j \neq J$ and y_J=1. In this case we can state

$$X_i = I_i z_{Ji}^{td} \qquad or \qquad \mathbf{X} = \mathbf{I} \cap \mathbf{z}_J^{td} \ , \tag{4}$$

where $\mathbf{z}_J^{td} \equiv \left(z_{J1}^{td}, z_{J2}^{td}, \ldots z_{JM}^{td} \right)$. This top-down template will be compared with the original input pattern $\mathbf{I}$ according to a predetermined *vigilance* criterion, tuned by a *vigilance parameter* $0 < \rho \leq 1$, so that two alternatives may occur:

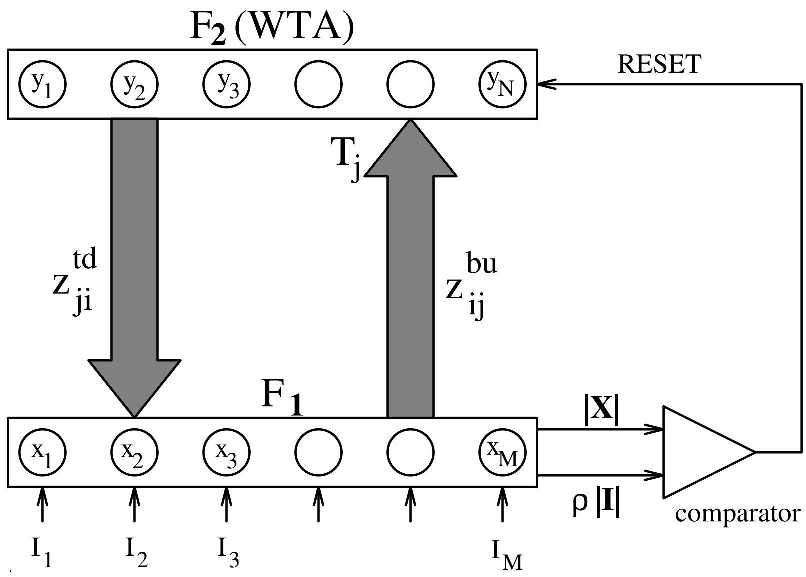

Fig. 1: Simplified block diagram of the architecture of a *Type-3* ART 1 system

---

1. Bottom-up weights z_{ij}^{bu} may take any real value in the interval [0,K], where $K = \frac{L}{L-1+M}$, and $L > 1$ [Carpenter, 1987a].
2. In principle, layer F_2 is not restricted to act as a Winner-Take-All network. Contrast enhancement is another possible choice [Carpenter, 1987a].
3. In the *Fast Learning (Type-3)* model top-down weights z_{ji}^{td} may take only the values '0' or '1'.

a)If $\rho|\mathbf{I}| \le \left|\mathbf{I} \cap \mathbf{z}_J^{td}\right|$, the active category J is accepted, and the system weights will be updated to incorporate this new knowledge[4].

b)If $\rho|\mathbf{I}| > \left|\mathbf{I} \cap \mathbf{z}_J^{td}\right|$, the active category J is not valid for the given value of the *vigilance parameter* ρ. In this case y_J will be deactivated (reset) making $T_J = 0$, so that another y_j node will become active through the Winner-Take-All action of the F_2 layer.

Learning takes place when an active F_2 node is accepted by the vigilance criterion. The weights will be updated according to the following algebraic equations,

$$
\begin{aligned}
z_{iJ}^{bu}(new) &= \frac{L}{L-1+\left|\mathbf{z}_J^{td}(old) \cap \mathbf{I}\right|} X_i = \frac{L}{L-1+\left|\mathbf{z}_J^{td}(old) \cap \mathbf{I}\right|} I_i z_{Ji}^{td}(old) \\
z_{Ji}^{td}(new) &= X_i = I_i z_{Ji}^{td}(old)
\end{aligned}
\tag{5}
$$

or, using vector notation,

$$
\begin{aligned}
\mathbf{z}_J^{bu}(new) &= \frac{L\mathbf{I} \cap \mathbf{z}_J^{td}(old)}{L-1+\left|\mathbf{I} \cap \mathbf{z}_J^{td}(old)\right|} \\
\mathbf{z}_J^{td}(new) &= \mathbf{I} \cap \mathbf{z}_J^{td}(old)
\end{aligned}
\tag{6}
$$

where $L > 1$ is a constant parameter. Note that only the weights of the connections incident to the winning F_2 node y_J are updated. Therefore, the operation of the *Type-3* (or *Fast Learning*) implementation of the ART 1 architecture is described by the algorithm depicted in Fig. 2(a).

From a hardware implementation point of view, one of the first issues that comes into consideration is that there are two templates of weights to be built. The set of bottom-up weights z_{ij}^{bu}, each of which must store a real value belonging to the interval [0,K], and the set of top-down weights z_{ji}^{td}, each of which stores either the value '0' or '1'. The physical implementation of the bottom-up template memory presents the first hardware difficulty because the weights need either an analog or a digital memory with sufficient bits per weight so that the digital discretization does not affect the system performance. However, it can be seen from eqs. (6) that the bottom-up set $\{z_{ij}^{bu}\}$ and the top-down set $\{z_{ji}^{td}\}$ contain the same information: each of these sets can be fully computed by knowing the other set. The bottom-up set $\{z_{ij}^{bu}\}$ is a normalized version of the top-down set $\{z_{ji}^{td}\}$. Therefore, from a hardware implementation point of view, it would be desirable to implement physically only a binary valued set (one bit per weight) and introduce the normalization of the bottom-up weights during the computation of $\{T_j\}$. This way, the two sets $\{z_{ij}^{bu}\}$ and $\{z_{ji}^{td}\}$ can be substituted by a single binary valued set $\{z_{ij}\}$, and eq. (1) modified to take into account the normalization effect of the original bottom-up weights,[5]

---

4. The notation $|\mathbf{a}|$ represents the cardinality of vector $\mathbf{a}$, i.e., $|\mathbf{a}| = \sum_i |a_i|$.

5. This type of modification is employed in the Fuzzy-ART model [Carpenter, 1991a], which operates with analog patterns, instead of binary ones. Making Fuzzy-ART to work with binary patterns results in ART 1 behavior, but using only one set of weights, similar to the system described in this paper.

$$T_j = \frac{L|\mathbf{I} \cap \mathbf{z}_j|}{L - 1 + |\mathbf{z}_j|} = \frac{L\sum_{i=1}^{M} z_{ij} I_i}{L - 1 + \sum_{i=1}^{M} z_{ij}} . \tag{7}$$

Considering this minor "implementation" modification, the algorithm of Fig. 2(a) would be transformed into that depicted in Fig. 2(b). The system level performance of the algorithms described by Fig. 2(a) and (b) is identical. There is no difference in the behavior between the two diagrams, and the one in Fig. 2(b) offers more attractive features from a hardware (as well as software) implementation point of view. For the remainder of this paper we will consider the original ART 1 *Type-3* architecture as described by the algorithm of Fig. 2(b).

However, in Fig. 2(b), an extra division operation, $T_j = (L|\mathbf{I} \cap \mathbf{z}_j|) / (L - 1 + |\mathbf{z}_j|)$, needs to be performed for each node in the F_2 layer. This is an expensive hardware operation and would probably constitute a performance bottleneck in the overall system for both analog and digital circuit implementations. If possible, it would be very desirable to avoid this division operation. The main idea of this paper is precisely to substitute this

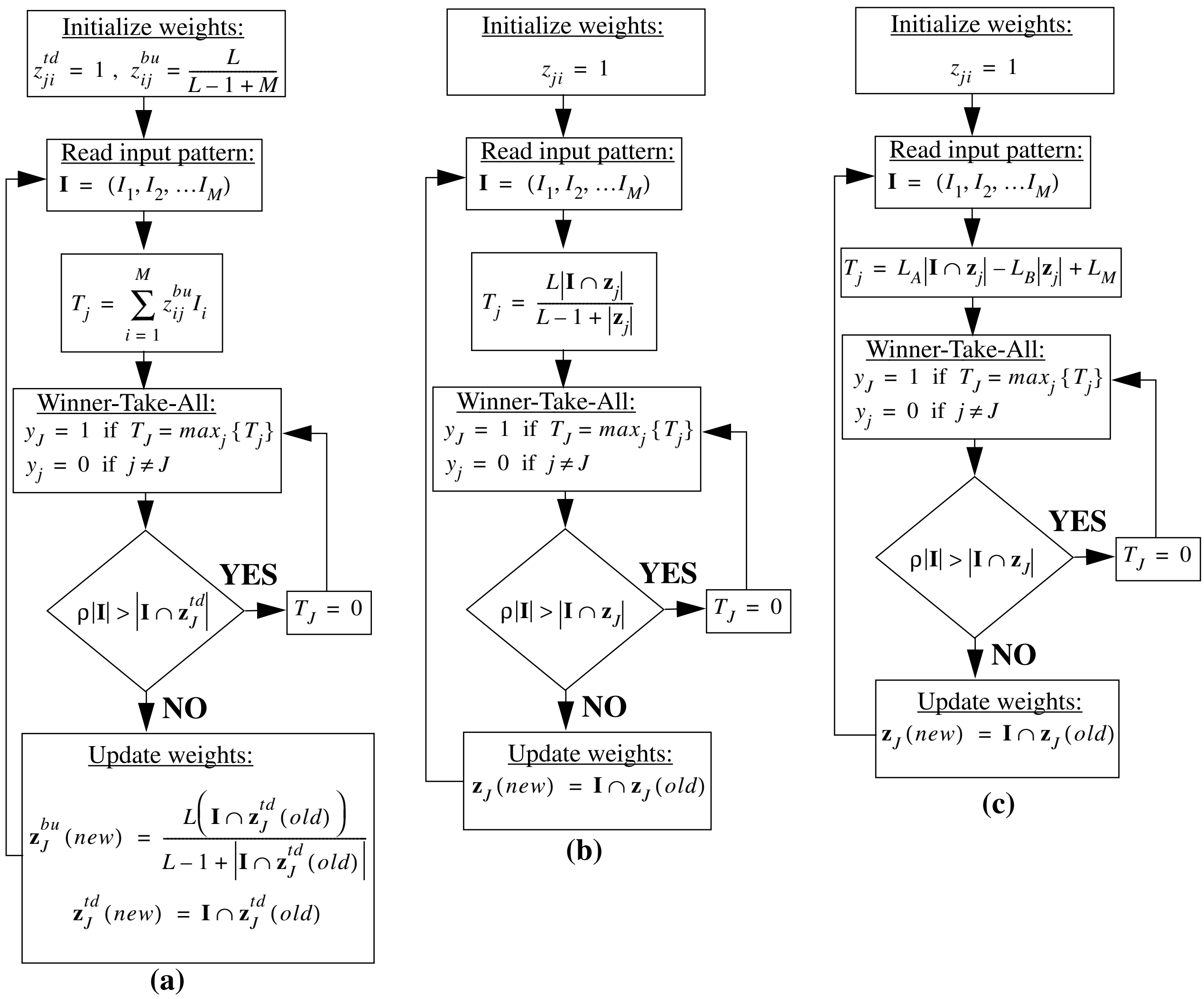

Fig. 2: *Type-3* implementation algorithms of the ART 1 architecture: (a) original ART 1, (b) ART 1 with a single binary valued weights template, (c) and VLSI-friendly ART 1_m

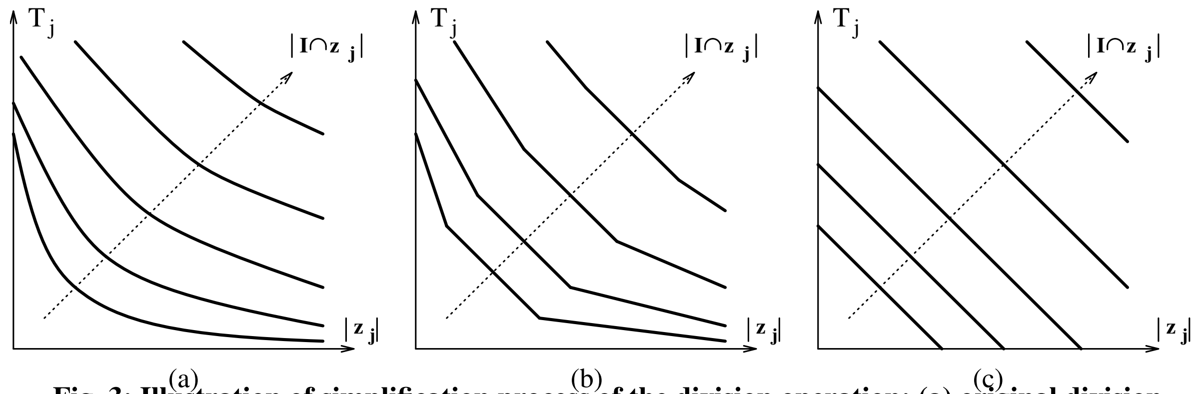

Fig. 3: Illustration of simplification process of the division operation: (a) original division operation, (b) piece-wise linear approximation, (c) linear approximation

division operation by another, less expensive one, and, although this results in a system with a slightly different behavior, we will show that it preserves all the computational properties of the original ART 1 algorithm.

Fig. 3(a) shows the curves that represent the division operation of eq. (7). A first simplification could be to substitute these curves by a piece-wise linear approximation as shown in Fig. 3(b). Such an approximation still presents some hardware difficulties and could also limit the performance of the overall system. A more drastic simplification would be to substitute the original operation by the operation represented by the set of curves of Fig. 3(c). Mathematically, the division operation has been substituted by a subtraction operation[6],

$$T_j = L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M \ , \tag{8}$$

where L_A and L_B are positive parameters that play the role of the original L (and L–1) parameter. As we will see in the next Section, the condition $L_A > L_B$ must be imposed for proper system operation. $L_M > 0$ is a constant parameter needed[7] to ensure that $T_j \geq 0$ for all possible values of $|\mathbf{I} \cap \mathbf{z}_j|$ and $|\mathbf{z}_j|$.

Replacing a division operation with a subtraction one is a very important hardware simplification with significant performance improvement potential. Fig. 2(c) shows the final *Type-3* ART 1_m algorithm, the object of this paper. In the next sections, we will try to show that the price paid for this drastic simplification, although it yields a system with slightly different input-output behavior, is insignificant since all the computational properties of the original ART 1 architecture are preserved.

It is worth mentioning here that substituting a division operation by a subtraction one means a significant performance boost from a hardware implementation point of view. Implementing physically division operators

---

6. During the writing of this paper, similar T_j functions (also called *distances* or *choice functions*) have been proposed by other authors for Fuzzy-ART. Since ART 1 can be considered a particular case of Fuzzy-ART when the input patterns are binary, Fuzzy-ART *choice functions* can also be used for ART 1. In the Appendix we show how these other *choice functions* also yield to ART 1 architectures that preserve as well all the original computational properties. However, the *choice function* purpose of this paper is computationally less expensive and is easier to implement in hardware.
7. In reality, parameter L_M has been introduced for hardware reasons [Serrano, 1994, 1996]. In a software ART 1_m implementation parameter L_M can be ignored.

in hardware constraints significantly the whole system design and imposes limitations on the overall system performance.

In the case of digital hardware, a division circuit can be built using either sequential techniques or large size higher speed special purpose circuits [Cavanagh, 1985]. Sequential techniques use simpler hardware but are slower, while a dedicated circuit is very large compared to the former and requires much more power consumption. As an example, and for a sequential type division circuit, in order to realize the following division

$$T_j = \frac{L|\mathbf{I} \cap \mathbf{z}_j|}{L - 1 + |\mathbf{z}_j|}, \tag{9}$$

q addition/substractions operation would be needed, where q is the number of bits needed for the result of the division. If, for example, there are $M = 1000$ nodes in the F_1 layer, numerator and denominator in eq. (9) should be represented by 10-bit words. If, for a given input $\mathbf{I}$, we want to differentiate between two terms T_{j_1} and T_{j_2} whose respective templates $\mathbf{z}_{j_1}$ and $\mathbf{z}_{j_2}$ differ in one bit, the F_2 layer (WTA) would need to resolve

$$|\Delta T_{j_1 j_2}|_{min} = \left| \frac{L|\mathbf{I} \cap \mathbf{z}_{j_1}|}{L - 1 + |\mathbf{z}_{j_1}|} - \frac{L|\mathbf{I} \cap \mathbf{z}_{j_2}|}{L - 1 + |\mathbf{z}_{j_2}|} \right|_{min}. \tag{10}$$

The worst case occurs when $|\mathbf{z}_{j_1}| = |\mathbf{I} \cap \mathbf{z}_{j_1}| = M$, $|\mathbf{z}_{j_2}| = |\mathbf{I} \cap \mathbf{z}_{j_2}| = M - 1$. In this case

$$|\Delta T_{j_1 j_2}|_{min} = \left| \frac{L(L-1)}{(L-1+M)\,(L-2+M)} \right|. \tag{11}$$

A reasonable minimum value for L is 1.01. Therefore, if $M = 1000$ then $|\Delta T_{j_1 j_2}|_{min} \approx 10^{-8}$. On the other hand, it is easy to see that $|\Delta T_{j_1 j_2}|_{max}$ is close to but less than one. Consequently, for each T_j a dynamic range of

$$\frac{|T_j|_{max}}{|T_j|_{min}} = 10^8 \tag{12}$$

is needed. Such dynamic range requires a q=27 bit representation. Thus, for each division operation we need to realize 27 10-bit addition/subtractions. Furthermore, the WTA in the F_2 layer would need to choose the maximum among N 27-bit words. On the other hand, if the ART 1_m algorithm is used, instead of the $N \times 24$ 11-bit addition/subtractions, we need only to realize N 11-bit subtractions, and the WTA has to choose the maximum among N 11-bit words.

In the case of analog hardware, there are ways to implement the division operation with compact dedicated circuits [Bult, 1987], [Sánchez-Sinencio, 1989], [Gilbert, 1990], [Sheingold, 1976], but they usually suffer from low signal-to-noise ratios, limited signal range, noticeable distortion, or require bipolar devices which are available for more expensive VLSI technologies. In any case, the performance of the overall ART system would be limited by the lower performance of the division operators. If the divison operators are eliminated the performance of the system would be limited by other operators which, for the same VLSI technology, render considerable better performance figures. Furthermore, in the case of analog current mode signal processing [Serrano, 1996], the addition and subtraction of currents does not need any physical components. Consequently, by eliminating the need of signal division, the circuitry is dramatically simplified and its performance drastically improved.

III. On the Computational Equivalence of the Original and the Modified Models

Throughout the original ART 1 paper [Carpenter, 1987a], Carpenter and Grossberg provide rigorous demonstrations of the computational properties of the ART 1 architecture. Some of these properties are concerned with *Type-1* and *Type-2* operations of the architecture, but most refer to the *Type-3* model operation. From a functional point of view, i.e., when looking at the ART 1 system as a black box regardless of the details of its internal operations, the system level computational properties of ART 1 are fully contained in its *Fast-Learning* or *Type-3* model. The theorems and demonstrations given by Carpenter and Grossberg [Carpenter, 1987a] relating to *Type-1* and *Type-2* models of the system only ensure proper *Type-3* behavior. The purpose of this Section is to demonstrate that the modified *Type-3* model developed during the previous Section preserves all the *Type-3* computational properties of the original ART 1 architecture. The only functional difference between ART 1 and ART 1_m, is the way the terms T_j are computed before competing in the Winner-Take-All block. Therefore, the original properties and demonstrations that are not affected by the terms T_j will be automatically preserved. Such properties are, for example, the *Self-Scaling* property and the *Variable Coarseness* property tuned by the *Vigilance Parameter*. But there are other properties which are directly affected by the way the terms T_j are computed: *Subset and Superset Direct Access, Stable Category Learning, Biasing the Network to form New Categories*, and the properties consequent of the theorems in the original ART 1 paper [Carpenter, 1987a]. In the remainder of this Section we will show that these properties remain in the ART 1_m architecture.

Let us define a few concepts before demonstrating that the original computational properties are preserved.

a) *Direct Access:* an input pattern $\mathbf{I}$ is said to have *Direct Access* to a learned category y_j if this category is the first one selected by the Winner-Take-All F_2 layer and is accepted by the *vigilance subsystem*, so that no reset occurs.

b) *Subset Template:* an input pattern $\mathbf{I}$ is said to be a *Subset Template* of a learned category $\mathbf{z}_j \equiv (z_{1j}, z_{2j}, \ldots z_{Mj})$ if $\mathbf{I} \subset \mathbf{z}_j$. Formally,

$$\begin{aligned} z_{ij} = 0 \Rightarrow I_i = 0 \qquad & \forall i = 1, \ldots M, \\ I_i = 1 \Rightarrow z_{ij} = 1 \qquad & \forall i = 1, \ldots M, \\ \text{there are some values of } i \text{ such that} \quad I_i = 0 \quad & \text{and} \quad z_{ij} = 1. \end{aligned} \tag{13}$$

c) *Superset Template:* an input pattern $\mathbf{I}$ is said to be a *Superset Template* of a learned category $\mathbf{z}_j$ if $\mathbf{z}_j \subset \mathbf{I}$.

d) *Mixed Template:* $\mathbf{z}_j$ and $\mathbf{I}$ are said to be mixed templates if neither $\mathbf{I} \subset \mathbf{z}_j$ nor $\mathbf{z}_j \subset \mathbf{I}$ are satisfied, and $\mathbf{I} \neq \mathbf{z}_j$.

e) *Uncommitted node:* an F_2 node y_j is said to be uncommitted if all its weights z_{ij} ($i = 1, \ldots M$) preserve their initial value ($z_{ij} = 1$), i.e., node y_j has not yet been selected to represent any learned category.

A. Direct Access to Subset and Superset Patterns

Suppose that a learning process has produced a set of categories in the F_2 layer. Each category y_j is characterized by the set of weights that connect node y_j in the F_2 layer to all nodes in the F_1 layer, i.e., $\mathbf{z}_j \equiv (z_{1j}, z_{2j}, \ldots, z_{Mj})$. Suppose that two of these categories, y_{j_1} and y_{j_2}, are such that $\mathbf{z}_{j_1} \subset \mathbf{z}_{j_2}$ ($\mathbf{z}_{j_1}$ is a subset template of $\mathbf{z}_{j_2}$). Now consider two input patterns $\mathbf{I}^{(1)}$ and $\mathbf{I}^{(2)}$ such that,

$$
\begin{aligned}
\mathbf{I}^{(1)} &= \mathbf{z}_{j_1} \equiv (z_{1j_1}, z_{2j_1}, \ldots z_{Mj_1}) \quad , \\
\mathbf{I}^{(2)} &= \mathbf{z}_{j_2} \equiv (z_{1j_2}, z_{2j_2}, \ldots z_{Mj_2}) \quad .
\end{aligned}
\tag{14}
$$

The *Direct Access to Subset and Superset* property assures that input $\mathbf{I}^{(1)}$ will have *Direct Access* to category y_{j_1} and that input $\mathbf{I}^{(2)}$ will have *Direct Access* to category y_{j_2}.

Proof:

If pattern $\mathbf{I}^{(1)}$ is given as the input pattern we will have

$$
\begin{aligned}
T_{j_1} &= L_A \sum_{i=1}^{M} I_i^{(1)} z_{ij_1} - L_B |\mathbf{z}_{j_1}| + L_M = L_A |\mathbf{I}^{(1)}| - L_B |\mathbf{I}^{(1)}| + L_M \quad , \\
T_{j_2} &= L_A \sum_{i=1}^{M} I_i^{(1)} z_{ij_2} - L_B |\mathbf{z}_{j_2}| + L_M = L_A |\mathbf{I}^{(1)}| - L_B |\mathbf{I}^{(2)}| + L_M \quad .
\end{aligned}
\tag{15}
$$

Since $|\mathbf{I}^{(1)}| < |\mathbf{I}^{(2)}|$, it follows that (remember L_B>0) $T_{j_1} > T_{j_2}$. If pattern $\mathbf{I}^{(2)}$ is presented at the input layer of the network, it would be,

$$
\begin{aligned}
T_{j_1} &= L_A \sum_{i=1}^{M} I_i^{(2)} z_{ij_1} - L_B |\mathbf{z}_{j_1}| + L_M = L_A |\mathbf{I}^{(1)}| - L_B |\mathbf{I}^{(1)}| + L_M \quad , \\
T_{j_2} &= L_A \sum_{i=1}^{M} I_i^{(2)} z_{ij_2} - L_B |\mathbf{z}_{j_2}| + L_M = L_A |\mathbf{I}^{(2)}| - L_B |\mathbf{I}^{(2)}| + L_M \quad .
\end{aligned}
\tag{16}
$$

In order to guarantee that $T_{j_2} > T_{j_1}$, the condition

$$
L_A > L_B \tag{17}
$$

must be satisfied.

B. Direct Access by perfectly learned patterns (Theorem 1 of original ART 1):

This theorem, when adapted to a *Type-3* implementation, would state the following:

`An input pattern` $\mathbf{I}$ `has direct access to a node` y_J `which has perfectly learned` $\mathbf{I}$ `.`

Proof:

In the case of the ART 1_m algorithm, in order prove that $\mathbf{I}$ has direct access to y_J we need to show that: (i) y_J is the first F_2 node to be chosen, (ii) y_J is accepted by the vigilance criterion, and (iii) y_J remains active as learning occurs[8].

To prove property (i), we must establish that, at the start of each trial, $T_J > T_j$ for all $j \neq J$. Since $\mathbf{z}_J = \mathbf{I}$, we need to prove

$$T_J = L_A|\mathbf{I}| - L_B|\mathbf{I}| > L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| = T_j \ . \tag{18}$$

Suppose first that $|\mathbf{z}_j| > |\mathbf{I}|$. Since $|\mathbf{I}| \geq |\mathbf{I} \cap \mathbf{z}_j|$ is always true, then eq. (18) is satisfied,

$$T_J = L_A|\mathbf{I}| - L_B|\mathbf{I}| \geq L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{I}| > L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| = T_j . \tag{19}$$

Suppose that $|\mathbf{z}_j| \leq |\mathbf{I}|$. Then, since $\mathbf{z}_j \neq \mathbf{I}$, it follows that $|\mathbf{I}| > |\mathbf{I} \cap \mathbf{z}_j|$. Finally, since $|\mathbf{z}_j| \geq |\mathbf{I} \cap \mathbf{z}_j|$ is always true, it follows that,

$$T_J = L_A|\mathbf{I}| - L_B|\mathbf{I}| > L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{I} \cap \mathbf{z}_j| \geq L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| = T_j \ . \tag{20}$$

Property (ii) is directly satiesfied because,

$$|\mathbf{I} \cap \mathbf{z}_j| = |\mathbf{I}| \geq \rho|\mathbf{I}| \qquad , \qquad \forall \rho \in [0, 1] \ . \tag{21}$$

Finally, property (iii) also holds, because after node y_J is selected as the winning category, its weight template $\mathbf{z}_j$ will remain unchanged (because $\mathbf{z}_J(new) = \mathbf{I} \cap \mathbf{z}_j(old) = \mathbf{I} = \mathbf{z}_j(old)$), and consequently the inputs to the F_2 layer T_j will remain unchanged.

C. Stable Choices in STM (Theorem 2 of original ART 1):

Whenever an input pattern $\mathbf{I}$ is presented for the first time to the ART 1 system, a set of $\{T_j\}$ values is formed that compete in the Winner-Take-All F_2 layer. The winner may be reset by the *vigilance subsystem*, and a new winner appears that may also be reset, and so on until a final winner is accepted. During this search process, the T_j values that led to earlier winners are set to zero. Let us call O_j the values of T_j at the beginning of the search process, i.e., before any of them is set to zero by the vigilance subsystem. Theorem 2 of the original ART 1 architecture states:

---

8. In the original ART 1 paper it also shown that read out of the top-down template does not deactivate node y_J as the winning node. This is because there the proof was developed for a *Type-1* implementation where activation of an F_2 node results in a change of T_j terms through the influence of the top-down connections.

> Suppose that an F_2 node y_J is chosen for STM storage instead of another node y_j because $O_J > O_j$. Then read-out of the top-down template preserves the inequality $T_J > T_j$ and thus confirms the choice of y_J by the bottom-up filter.

This theorem has only sense for a *Type-1* implementation, because there, as a node in the F_2 layer activates,the initial values of T_j (immediately after presenting an input pattern $\mathbf{I}$) may be altered through the top-down "*feed-back*" connections. In a *Type-3* description (see Fig. 2) the initial terms T_j remain unchanged, independently of what happens in the F_2 layer. Therefore, this theorem is implicitely satisfied.

D. Initial Filter Values determine Search Order (Theorem 3 of original ART 1):

Theorem 3 of the original ART 1 architecture states that (page 92 of [Carpenter, 1987a]):

> The Order Function ($O_{j_1} > O_{j_2} > O_{j_3} > \ldots$) determines the order of search no matter how many times F_2 is reset during a trial.

The proof is the same for the ART 1 and the ART 1_m (both *Type-3*) implementations[9]. If T_{j_1} is reset by the *vigilance subsystem*, the values of $T_{j_2}, T_{j_3}, \ldots$ will not change. Therefore, the new order sequence is $O_{j_2} > O_{j_3} > \ldots$ and the original second largest value O_{j_2} will be selected as the winner. If T_{j_2} is now set to zero, O_{j_3} is the next winner, and so on.

This Theorem, although trivial in a *Type-3* implementation, has more importance in a *Type-1* description where the process of selecting and shutting down a winner alters all values T_j [Carpenter, 1987a].

E. Learning on a Single Trial (Theorem 4 of original ART 1):

This theorem (page 93 of [Carpenter, 1987a]) states the following, assuming a *Type-3* implementation is being considered[10]:

> Suppose that an F_2 winning node y_J is accepted by the vigilance subsystem. Then the LTM traces z_{ij} change in such a way that T_J increases and all other T_j remain constant, thereby confirming the choice of y_J. In addition, the set $\mathbf{I} \cap \mathbf{z}_J$ remains constant during learning, so that learning does not trigger reset of y_J by the vigilance subsystem.

Proof:

In this case, if y_J is the winning category accepted by the *vigilance subsystem*, from eq. (8) we obtain

$$T_J = L_A|\mathbf{I} \cap \mathbf{z}_J| - L_B|\mathbf{z}_J| + L_M \ . \tag{22}$$

9. However, note that the resulting ordering $\{j_1, j_2, j_3, \ldots\}$ may differ for the original and the modified architecture.
10. A more sophisticated demonstration for this theorem is provided in the original ART 1 paper [Carpenter, 1987a]. This is because the demonstration is performed for a *Type-1* description of ART 1.

The update rule is

$$\mathbf{z}_J(new) = \mathbf{I} \cap \mathbf{z}_J(old) \quad , \tag{23}$$

and the new T_J value is given by,

$$\begin{aligned} T_J(new) &= L_A|\mathbf{I} \cap \mathbf{z}_J(new)| - L_B|\mathbf{z}_J(new)| + L_M = \\ &= L_A|\mathbf{I} \cap \mathbf{I} \cap \mathbf{z}_J(old)| - L_B|\mathbf{I} \cap \mathbf{z}_J(old)| + L_M = L_A|\mathbf{I} \cap \mathbf{z}_J(old)| - L_B|\mathbf{I} \cap \mathbf{z}_J(old)| + L_M \geq \\ &\geq L_A|\mathbf{I} \cap \mathbf{z}_J(old)| - L_B|\mathbf{z}_J(old)| + L_M = T_J(old) \quad . \end{aligned} \tag{24}$$

Therefore, learning confirms the choice of y_J, and by eq. (23) the set $\mathbf{I} \cap \mathbf{z}_J$ remains constant.

F. Stable Category Learning (Theorem 5 of original ART 1):

Suppose an arbitrary list (finite or infinite) of binary input patterns is presented to an ART 1_m system. Each template set $\mathbf{z}_j \equiv (z_{1j}, z_{2j}, \ldots z_{Mj})$ is updated every time category y_j is selected by the Winner-Take-All F_2 layer and accepted by the vigilance subsystem. Some times template $\mathbf{z}_j$ may be changed, and others it may remain unchanged. Let us call the times $\mathbf{z}_j$ suffers a change $t_1^{(j)} < t_2^{(j)} < \ldots < t_{r_j}^{(j)}$. Since the vector (or template) $\mathbf{z}_j$ of a committed node has M components (of which, at the most, $M-1$ are set to '1'), and by eq. (23) each component can only change from '1' to '0' but not from '0' to '1', it follows that template $\mathbf{z}_j$ can, at the most, suffer $M-1$ changes,

$$r_j \leq M - 1 \quad . \tag{25}$$

Since template $\mathbf{z}_j$ will remain unchanged after time $t_{r_j}^{(j)}$, it is concluded that the complete LTM memory will suffer no change after time

$$t_{learn} = max_j \{t_{r_j}^{(j)}\} \quad . \tag{26}$$

If there is a finite number of nodes in the F_2 layer t_{learn} has a finite value, and thus learning is completed after a finite number of time steps.

This is true for both the ART 1 and the ART 1_m architectures. Therefore, the following theorem (page 95 of [Carpenter, 1987a]) is valid for the two algorithms:

> In response to an arbitrary list of binary input patterns, all LTM traces $z_{ij}(t)$ approach limits after a finite number of learning trials. Each template set $\mathbf{z}_j$ remains constant except for at most $M-1$ times $t_1^{(1)} < t_2^{(2)} < \ldots < t_{r_j}^{(j)}$ at which it progressively loses elements, leading to the
>
> Subset Recoding Property: $\mathbf{z}_j(t_1^{(j)}) \supset \mathbf{z}_j(t_2^{(j)}) \supset \ldots \supset \mathbf{z}_j(t_{r_j}^{(j)})$. (27)
>
> The LTM traces $z_{ij}(t)$ such that $i \notin \mathbf{z}_j(t_1^{(j)})$ decrease to zero. The LTM traces $z_{ij}(t)$ such that $i \in \mathbf{z}_j(t_{r_j}^{(j)})$ remain always at '1'. The LTM traces such that $i \in \mathbf{z}_j(t_{r_k}^{(j)})$ but $i \notin \mathbf{z}_j(t_{r_{k+1}}^{(j)})$ stay at '1' for times $t \leq t_k^{(1)}$ but will change to and stay at '0' for times $t \geq t_{k+1}^{(j)}$.

G. Direct Access after Learning Stabilizes (Theorem 6 of original ART 1):

Assuming F_2 has a finite number of nodes, the present theorem (page 98 of [Carpenter, 1987a]) states the following:

> After recognition learning has stabilized in response to an arbitrary list of binary input patterns, each input pattern $\mathbf{I}$ either has direct access to the node y_j which possesses the largest subset template with respect to $\mathbf{I}$, or $\mathbf{I}$ cannot be coded by any node of F_2. In the latter case, F_2 contains no uncommitted nodes.

Proof:

Since learning has already stabilized $\mathbf{I}$ can be coded only by a node y_j whose template $\mathbf{z}_j$ is a subset template with respect to $\mathbf{I}$. Otherwise, once y_j becomes active, the set $\mathbf{z}_j$ would contract to $\mathbf{z}_j \cap \mathbf{I}$, thereby contradicting the hypothesis that learning has already stabilized. Thus, if $\mathbf{I}$ activates any node other than one with a subset template, that node must be reset by the *vigilance subsystem*. For the remainder of the proof, let y_J be the first F_2 node activated by $\mathbf{I}$. We need to show that if $\mathbf{z}_J$ is a subset template, then it is the subset template with the largest O_J; and if it is not a subset template, then all subset templates activated on that trial will be reset by the vigilance subsystem:

$$|\mathbf{I} \cap \mathbf{z}_j| = |\mathbf{z}_j| < \rho|\mathbf{I}| \ . \tag{28}$$

If y_J and y_j are nodes with subset templates with respect to $\mathbf{I}$, then

$$O_j = L_A|\mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M < O_J = L_A|\mathbf{z}_J| - L_B|\mathbf{z}_J| + L_M \ . \tag{29}$$

Since $(L_A - L_B)\,|\mathbf{z}_j|$ is an increasing function of $|\mathbf{z}_j|$,

$$|\mathbf{z}_j| < |\mathbf{z}_J| \tag{30}$$

and,

$$R_j = \frac{|\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{I}|} = \frac{|\mathbf{z}_j|}{|\mathbf{I}|} < R_J = \frac{|\mathbf{I} \cap \mathbf{z}_J|}{|\mathbf{I}|} = \frac{|\mathbf{z}_J|}{|\mathbf{I}|} \ . \tag{31}$$

Therefore, if y_J is reset ($R_J < \rho$), all other nodes with subset templates will be reset ($R_j < \rho$).

Now suppose that y_J, the first activated node, does not have a subset template with respect to $\mathbf{I}$ ($|\mathbf{I} \cap \mathbf{z}_J| < |\mathbf{z}_J|$), but another node y_j with a subset template is activated in the course of search. We need to show that $|\mathbf{I} \cap \mathbf{z}_j| = |\mathbf{z}_j| < \rho|\mathbf{I}|$, so that y_j is reset. We know that,

$$O_j = (L_A - L_B)\,|\mathbf{z}_j| + L_M < O_J = L_A|\mathbf{I} \cap \mathbf{z}_J| - L_B|\mathbf{z}_J| + L_M < (L_A - L_B)\,|\mathbf{z}_J| + L_M \ , \tag{32}$$

which implies that $|\mathbf{z}_j| < |\mathbf{z}_J|$. Since y_J cannot be chosen, it must be reset by the *vigilance subsystem*, which means that $|\mathbf{I} \cap \mathbf{z}_J| < \rho|\mathbf{I}|$. Therefore,

$$L_A|\mathbf{z}_j| - L_B|\mathbf{z}_j| < L_A|\mathbf{I} \cap \mathbf{z}_J| - L_B|\mathbf{z}_J| < L_A\rho|\mathbf{I}| - L_B|\mathbf{z}_J| < L_A\rho|\mathbf{I}| - L_B|\mathbf{z}_j| \ , \tag{33}$$

which implies that

$$|\mathbf{I} \cap \mathbf{z}_j| = |\mathbf{z}_j| < \rho|\mathbf{I}| \; . \tag{34}$$

H. Search Order (Theorem 7 of original ART 1):

The conditions expressed in the original Theorem 7 must be changed to adapt this theorem to the ART 1_m architecture. The modified theorem states the following:

Suppose

$$\frac{L_A}{L_B} < \frac{M}{M-1} \; , \tag{35}$$

and that input pattern $\mathbf{I}$ satisfies

$$|\mathbf{I}| \le M-1 \; . \tag{36}$$

Then F_2 nodes are searched in the following order, if they are searched at all.

Subset templates with respect to $\mathbf{I}$ are searched first, in order of decreasing size. If the largest subset template is reset, then all subset templates are reset. If all subset templates have been reset and if no other learned templates exist, then the first uncommitted node to be activated will code $\mathbf{I}$. If all subset templates are searched and if there exist learned superset templates but no mixed templates, then the node with the smallest superset template will be activated next and will code $\mathbf{I}$. If all subset templates are searched and if both superset templates $\mathbf{z}_J$ and mixed templates $\mathbf{z}_j$ exist, then y_j will be searched before y_J if and only if

$$|\mathbf{z}_j| < |\mathbf{z}_J| \qquad \text{and} \qquad \frac{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{z}_J| - |\mathbf{z}_j|} < \frac{L_B}{L_A} . \tag{37}$$

If all subset templates are searched and if there exist mixed templates but no superset templates, then a node y_j with a mixed template will be searched before an uncommitted node y_J if and only if

$$L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M > T_J(\mathbf{I}, \text{t=0}) \; . \tag{38}$$

where $T_J(\mathbf{I}, \text{t=0}) = L_A \sum I_i z_{iJ}(0) - L_B \sum z_{iJ}(0) + L_M$. The proof has several parts:

a) First we show that a node y_J with a subset template ($\mathbf{I} \cap \mathbf{z}_J = \mathbf{z}_J$) is searched before any node y_j with a non-subset template. In this case,

$$O_j = L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M = |\mathbf{I} \cap \mathbf{z}_j|\left(L_A - L_B\frac{|\mathbf{z}_j|}{|\mathbf{I} \cap \mathbf{z}_j|}\right) + L_M \; . \tag{39}$$

Now, note that

$$\frac{|\mathbf{z}_j|}{|\mathbf{I} \cap \mathbf{z}_j|} > \frac{M}{M-1} \tag{40}$$

because[11]

$$\left.\frac{|\mathbf{z}_j|}{|\mathbf{I} \cap \mathbf{z}_j|}\right|_{min} = \left.\frac{|\mathbf{z}_j|}{|\mathbf{z}_j| - 1}\right|_{min} = \frac{M-1}{M-2} > \frac{M}{M-1} \ . \tag{41}$$

From eqs. (35), (39) and (41), it follows that

$$O_j < |\mathbf{I} \cap \mathbf{z}_j| L_B\left(\frac{L_A}{L_B} - \frac{M}{M-1}\right) + L_M < L_M . \tag{42}$$

On the other hand,

$$O_J = (L_A - L_B)\,|\mathbf{z}_J| + L_M > L_M \ . \tag{43}$$

Therefore,

$$O_J > O_j \ . \tag{44}$$

b) Subset templates are searched in order of decreasing size:

Suppose two subset templates of $\mathbf{I}$, $\mathbf{z}_J$ and $\mathbf{z}_j$ such that $|\mathbf{z}_J| > |\mathbf{z}_j|$. Then

$$O_J = (L_A - L_B)\,|\mathbf{z}_J| + L_M > (L_A - L_B)\,|\mathbf{z}_j| + L_M = O_j \ . \tag{45}$$

Therefore node y_J will be searched before node y_j. By eq. (45), if the largest subset template is reset, all other subset templates are reset as well.

c) Subset templates y_J are searched before an uncommitted node y_j:

$$\begin{aligned} O_j &= L_A|\mathbf{I}| - L_B M + L_M \le L_A(M-1) - L_B M + L_M = L_B\left(\frac{L_A}{L_B}(M-1) - M\right) + L_M < \\ &< L_B\left(\frac{M}{M-1}(M-1) - M\right) + L_M = L_M < (L_A - L_B)\,|\mathbf{z}_J| + L_M = O_J \ . \end{aligned} \tag{46}$$

Therefore, if all subset templates are searched and if no other learned template exists, an uncommitted node will be activated and code the input pattern.

---

11. We assume that y_j is not an uncommitted node ($|\mathbf{z}_j| < M$).

d) If all subset templates have been searched and there are learned superset templates but no mixed templates, the node with the smallest superset template y_J will be activated (and not an uncommitted node y_j) and code **I**:

$$O_J = L_A|\mathbf{I}| - L_B|\mathbf{z}_J| + L_M > L_A|\mathbf{I}| - L_B M + L_M = O_j \ . \tag{47}$$

If there is more than one superset template, the one with the smallest $|\mathbf{z}_J|$ will be activated. Since $|\mathbf{I} \cap \mathbf{z}_J| = |\mathbf{I}| \geq \rho|\mathbf{I}|$, there is no reset, and **I** will be coded.

e) If all subset templates have been searched, and there exist a superset template y_J and a mixed template y_j, then $O_j > O_J$ if and only if eq. (37) holds:

$$O_j - O_J = L_A(|\mathbf{I} \cap \mathbf{z}_j| - |\mathbf{I}|) + L_B(|\mathbf{z}_J| - |\mathbf{z}_j|) \ . \tag{48}$$

e.1) If eq. (37) holds:

$$O_j - O_J = L_A\left(\frac{L_B}{L_A} - \frac{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{z}_J| - |\mathbf{z}_j|}\right)(|\mathbf{z}_J| - |\mathbf{z}_j|) > 0 \ . \tag{49}$$

e.2) If $O_j > O_J$:

Assume first that $|\mathbf{z}_J| - |\mathbf{z}_j| < 0$. Then, by eq. (49), it has to be

$$\frac{L_B}{L_A} < \frac{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{z}_J| - |\mathbf{z}_j|} \ . \tag{50}$$

Since $L_A > L_B > 0$ it had to be $|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j| < 0$, which is false. Therefore, it must be that $|\mathbf{z}_J| - |\mathbf{z}_j| > 0$, and

$$\frac{L_B}{L_A} > \frac{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{z}_J| - |\mathbf{z}_j|} \ . \tag{51}$$

f) If all subset templates are searched, and if there are mixed templates but no superset templates, then a node y_j with a mixed template ($O_j = L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M$) will be searched before an uncommitted node y_J ($O_J = L_A|\mathbf{I}| - L_B M + L_M$) if and only if eq. (38) holds:

$$\begin{aligned} O_j - O_J &= L_A(|\mathbf{I} \cap \mathbf{z}_j| - |\mathbf{I}|) - L_B(|\mathbf{z}_j| - M) > 0 \Leftrightarrow \\ \Leftrightarrow L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M &> L_A|\mathbf{I}| - L_B M + L_M = T_J(\mathbf{I}, \mathrm{t{=}0}) \ . \end{aligned} \tag{52}$$

This completes the proof of the modified Theorem 7 for the ART 1_m architecture.

I. Biasing the Network towards Uncommitted Nodes:

In the original ART 1 architecture, choosing L large increases the network's tendency to choose uncommitted nodes in response to unfamiliar input patterns **I**. In the ART 1_m architecture, the same effect is observed when choosing L_A/L_B large. This can be understood through the following reasoning.

When an input pattern **I** is presented, an uncommitted node is chosen before a coded node y_j if

$$L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| < L_A|\mathbf{I}| - L_B M \ . \tag{53}$$

This inequality is equivalent to

$$\frac{L_A}{L_B} > \frac{M - |\mathbf{z}_j|}{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|} \ . \tag{54}$$

As the ratio L_A/L_B increases it is more likely that eq. (54) be satisfied, and hence uncommitted nodes are chosen before coded nodes, regardless of the *vigilance parameter* value ρ.

J. Remarks:

Even though this Section has shown that the computational properties of the original ART 1 system are preserved in the ART 1_m system, the response of both systems to an arbitrary list of training patterns will not be exactly the same. The main underlying reason for this difference is that the initial ordering

$$O_{j_1} > O_{j_2} > O_{j_3} > \dots \tag{55}$$

is not always exactly the same for both architectures. The next Section will study the differences between the two ART 1 systems.

IV. On the Functional Differences between Original and Modified Model

As stated previously, the difference in behavior between the ART 1 and ART 1_m models is caused by the different orderings of the terms of eq. (55). Assuming that both models, at a certain time, have identical weight templates $\{\mathbf{z}_j\}$, and the same input pattern $\mathbf{I}$ is given, eq. (55) has the following two formulations:

$$\begin{aligned} &\text{Original ART 1:} && \frac{|\mathbf{I} \cap \mathbf{z}_{j_1}|}{L - 1 + |\mathbf{z}_{j_1}|} > \frac{|\mathbf{I} \cap \mathbf{z}_{j_2}|}{L - 1 + |\mathbf{z}_{j_2}|} > \frac{|\mathbf{I} \cap \mathbf{z}_{j_3}|}{L - 1 + |\mathbf{z}_{j_3}|} > \dots \\ &\text{Modified ART 1:} && \frac{L_A}{L_B}|\mathbf{I} \cap \mathbf{z}_{l_1}| - |\mathbf{z}_{l_1}| > \frac{L_A}{L_B}|\mathbf{I} \cap \mathbf{z}_{l_2}| - |\mathbf{z}_{l_2}| > \dots \end{aligned} \tag{56}$$

where j_k might be different than l_k. The ordering resulting for the original ART 1 description is modulated by parameter $L > 1$. For example, if L is very large compared to all $|\mathbf{z}_j|$ terms, then the ordering depends exclusively on the values of $|\mathbf{I} \cap \mathbf{z}_j|$,

$$|\mathbf{I} \cap \mathbf{z}_{j_1}| > |\mathbf{I} \cap \mathbf{z}_{j_2}| > |\mathbf{I} \cap \mathbf{z}_{j_3}| > \dots \tag{57}$$

If L is very close to 1, then the ordering depends on the ratios,

$$\frac{|\mathbf{I} \cap \mathbf{z}_{j_1}|}{|\mathbf{z}_{j_1}|} > \frac{|\mathbf{I} \cap \mathbf{z}_{j_2}|}{|\mathbf{z}_{j_2}|} > \frac{|\mathbf{I} \cap \mathbf{z}_{j_3}|}{|\mathbf{z}_{j_3}|} > \ldots \tag{58}$$

Likewise, for the ART 1_m description, the ordering is modulated by a single parameter $\alpha = L_A/L_B > 1$. If α is extremely large, the situation in eq. (57) results. However, for α very close to 1, the ordering depends on the differences,

$$|\mathbf{I} \cap \mathbf{z}_{l_1}| - |\mathbf{z}_{l_1}| > |\mathbf{I} \cap \mathbf{z}_{l_2}| - |\mathbf{z}_{l_2}| > |\mathbf{I} \cap \mathbf{z}_{l_3}| - |\mathbf{z}_{l_3}| > \ldots \tag{59}$$

Obviously, the behavior of the two ART 1 descriptions will be identical for large values of L and α. However, moderate values of L and α are desired in practical ART 1 applications. On the other hand, it can be expected that the behavior will also tend to be similar for very high values of ρ: if ρ is very close to 1, each training pattern will form an independent category. However, different training patterns will cluster into a shared category for smaller values of ρ. Therefore, a very similar behavior between ART 1 and ART 1_m will be expected for high values of ρ, while more differences in behavior might be apparent for smaller values of ρ.

In order to compare the two algorithms' behavior, we have performed exhaustive simulations using randomly generated training patterns sets[12]. As an illustration of a typical case where the two algorithms produce different learned templates, Fig. 4 shows the evolution of the memory templates, for both the ART 1 and the ART 1_m algorithms, using a randomly generated training set of 10 patterns with 25 pixels each. Weight templates for original ART 1 are named $\mathbf{z}_j$, while for ART 1_m they are named $\mathbf{z}_j'$. The vigilance parameter was set to $\rho = 0.4$ for the original ART 1 $L = 5$, and for the ART 1_m $\alpha = 2$. In Fig. 4, boxed category templates are those that met the vigilance criterion and had the maximum T_j value. If the box is drawn with a continuous line, the correponding $\mathbf{z}_j$ template suffered modifications due to learning. If the box is drawn with dashed line, learning did not alter the corresponding $\mathbf{z}_j$ template. Both algorithms stabilized their weights in 2 training trials. Looking at the learned templates we can see that input patterns 4 and 5 clustered in the same category for both algorithms ($\mathbf{z}_4$ for original ART 1 and $\mathbf{z}'_3$ for ART 1_m). This also ocurred for patterns 6 and 8 ($\mathbf{z}_3$ and $\mathbf{z}'_2$) and for patterns 3, 9 and 10 ($\mathbf{z}_5$ and $\mathbf{z}'_5$). However, patterns 1, 2, and 7 did not cluster in the same way in the two cases. In the original ART 1 algorithm patterns 1 and 7 clustered into category $\mathbf{z}_1$, while pattern 2 remained independent in category $\mathbf{z}_2$. In the ART 1_m algorithm patterns 1 and 2 clustered together into category $\mathbf{z}'_1$, while pattern 7 remained independent in category $\mathbf{z}'_4$.

To measure a distance between the two templates $\mathbf{z}_j$ and $\mathbf{z}'_j$, let us use the Hamming distance between two binary patterns $\mathbf{a} \equiv (a_1, a_2, \ldots a_M)$ and $\mathbf{b} \equiv (b_1, b_2, \ldots b_M)$,

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{M} f_d(a_i, b_i) \quad , \tag{60}$$

12. For all simulations in this paper, randomly generated training patterns sets were obtained with a 50% probability for a pixel to be either '1' or '0'.

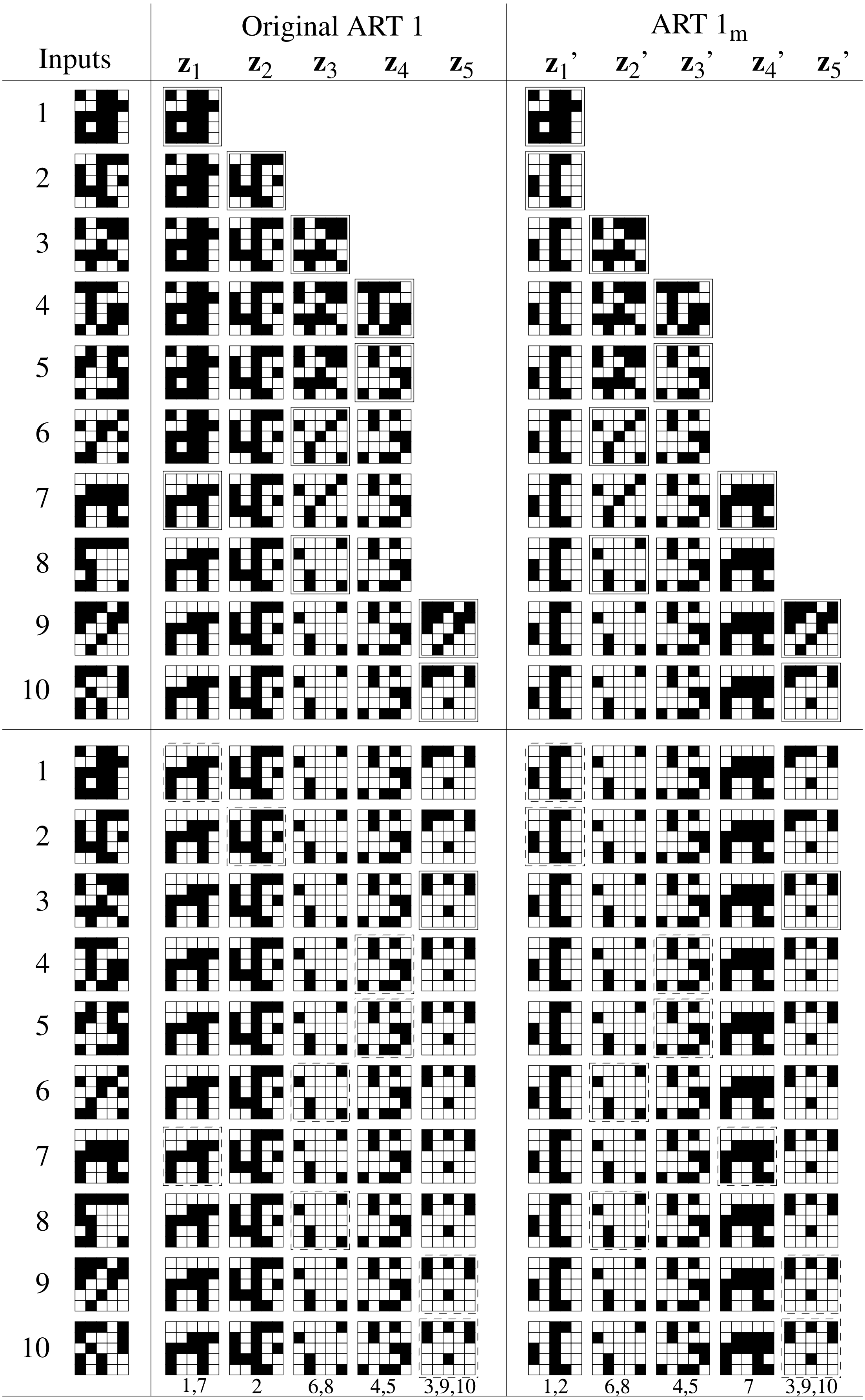

Fig. 4: Comparative Learning Example (ρ=0.4, L=5, α=2)

where

$$f_d(a_i, b_i) = \begin{cases} 0 & \text{if} \quad a_i = b_i \ , \\ 1 & \text{if} \quad a_i \neq b_i \ . \end{cases} \tag{61}$$

We can use this metric to define the distance between two sets of patterns $\{\mathbf{z}_j\}_{j=1}^{Q}$ and $\{\mathbf{z}'_j\}_{j=1}^{Q}$ as that which minimizes

$$\sum_{i=1}^{Q} d(\mathbf{z}_i, \mathbf{z}_{l_i}') \ . \tag{62}$$

For this purpose, the optimal ordering of indexes $(l_1, l_2, \ldots l_Q)$ must be found. In the case of Fig. 4 (where $Q = 5$), the distance D between the two learned patterns sets is given by,

$$D = d(\mathbf{z}_1, \mathbf{z}'_4) + d(\mathbf{z}_2, \mathbf{z}'_1) + d(\mathbf{z}_3, \mathbf{z}'_2) + d(\mathbf{z}_4, \mathbf{z}'_3) + d(\mathbf{z}_5, \mathbf{z}'_5) = 7 \ . \tag{63}$$

In general, we can define the distance between two patterns sets $\mathbf{A} = \{\mathbf{a}_j\}_{j=1}^{Q}$ and $\mathbf{B} = \{\mathbf{b}_j\}_{j=1}^{Q}$ as,

$$D(\mathbf{A}, \mathbf{B}) = \underset{\{l_1, l_2, \ldots l_Q\}}{min} \left[\sum_{i=1}^{Q} d(\mathbf{a}_i, \mathbf{b}_{l_i}) \right] \ . \tag{64}$$

In the case of Fig. 4, both algorithms produced the same number of learned categories. This does not always occur. For the case where a different number of categories results, we measured the distance between the two learned sets by adding as many uncommitted F_2 nodes to the set with less categories as necessary to equal the number of categories. An uncommitted category has all its pixels set to '1'. Thus, having a different number of committed nodes drastically increases the resulting distance, and is consequently a strong penalty.

We have repeated the simulation of Fig. 4 many times for different sets of randomly generated training patterns and sweeping the values of ρ, L, and α. For each combination of ρ, L, and α values, we repeated the simulation 100 times for different training patterns sets, and computed the average number of learned categories, learning trials, and distance between learned categories, as well as their corresponding standard deviations. Fig. 5 and Fig. 6 present the results of these simulations. Fig. 5(a) shows how the average number of learned categories changes with L (from 1.01 to 40) for different values of ρ, for the original ART 1. As ρ decreases, parameter L has more control on the average number of learned categories. Fig. 5(b) shows the standard deviation for the number of learned categories of Fig. 5(a). As the number of learned categories approaches the number of training patterns (10 in this case), standard deviation decreases. This happens for large values of L (independently of ρ) and for large values of ρ (independently of L). Fig. 5(c) and Fig. 5(d) show the same as Fig. 5(a) and Fig. 5(b) respectively, for the ART 1_m algorithm. As we can see, parameter α (swept from 1.01 to 5.0) of ART 1_m has more tuning power than parameter L of the original ART 1. On the other hand, ART 1_m presents a slightly higher standard deviation than the original ART 1. Nevertheless, the qualitative behavior of both

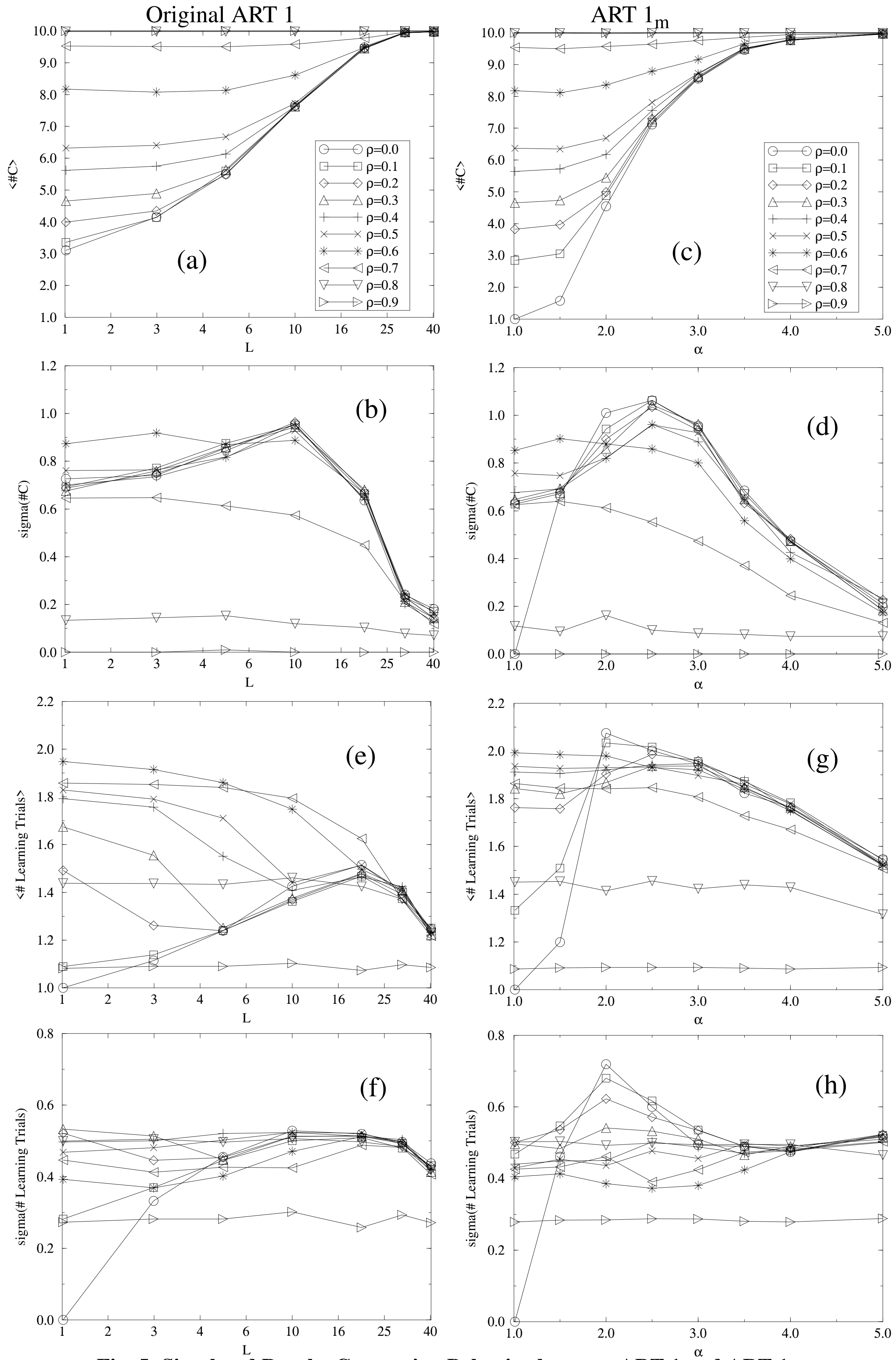

Fig. 5: Simulated Results Comparing Behavior between ART 1 and ART 1_m

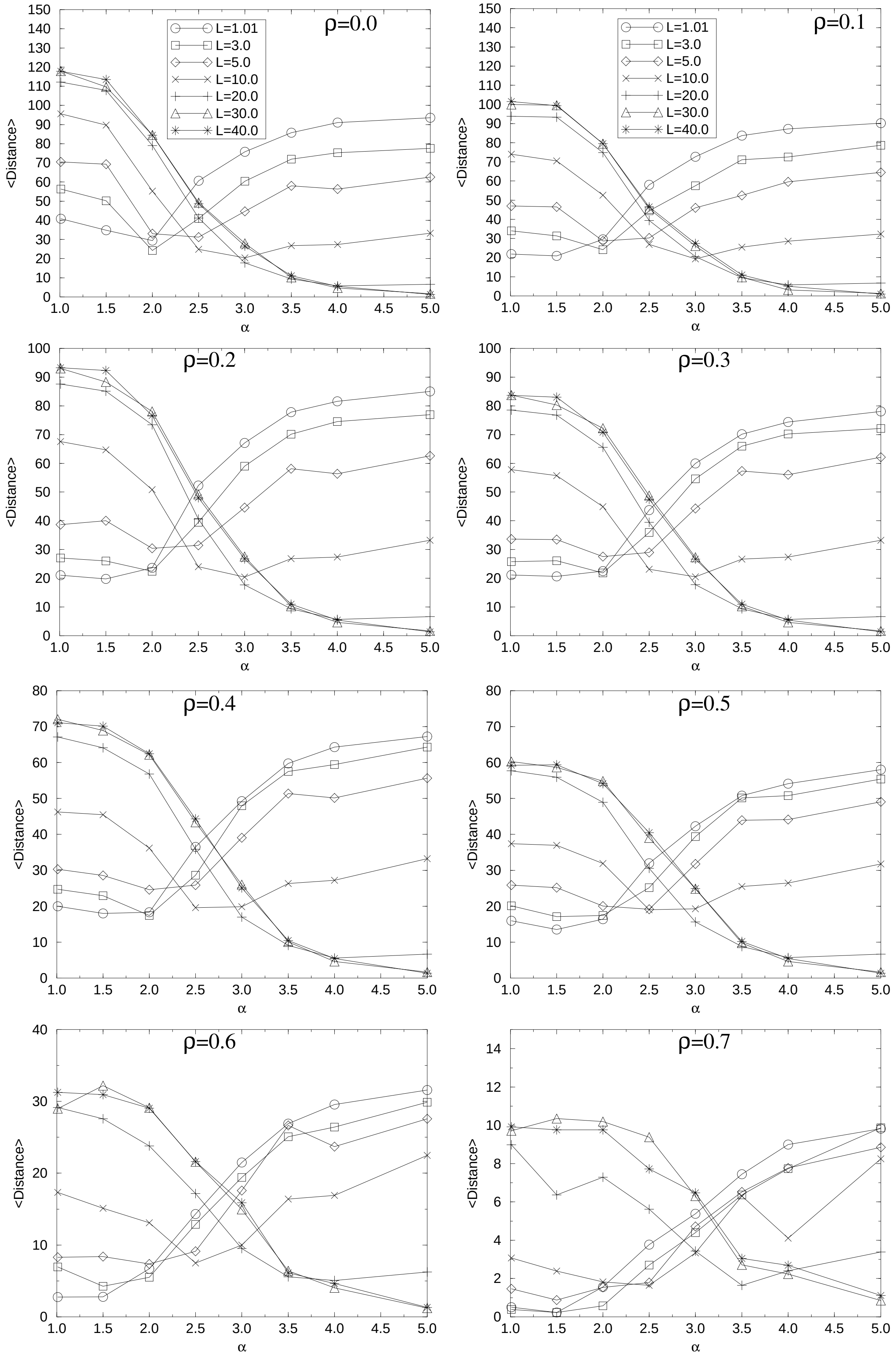

Fig. 6: Learned Categories Average Distances

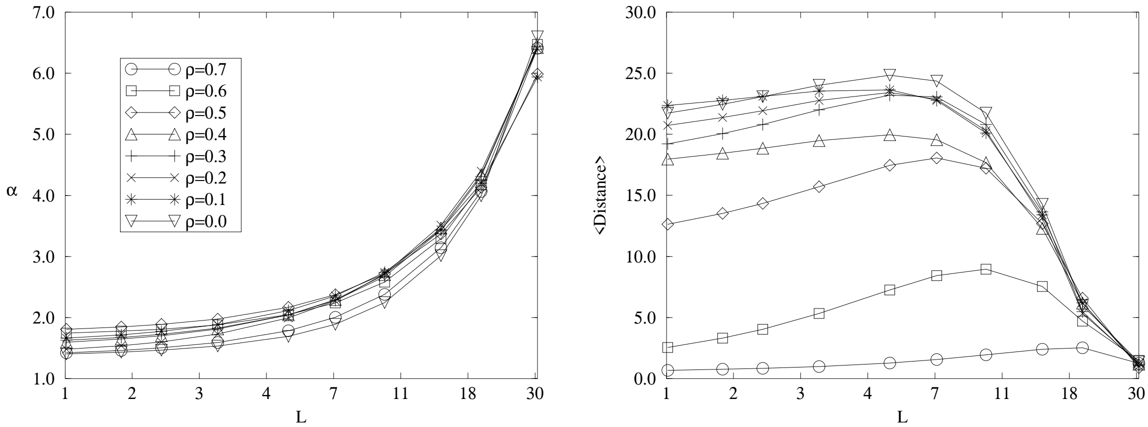

Fig. 7: Optimal parameters fit between ART 1 and ART 1_m

algorithms is similar. Fig. 5(e) and Fig. 5(f) show the average number of learning trials and their corresponding deviations, needed by the original ART 1 algorithm to stabilize its learned weights. Fig. 5(g) and Fig. 5(h) show the same for the ART 1_m algorithm. As we can see, the ART 1_m algorithm needs a slightly higher average number of learning trials to stabilize. Also, the standard deviation observed for the ART 1_m algorithm is slightly higher. Finally, Fig. 6 shows the resulting average distances (as defined by eq. (64)) between learned categories of the ART 1 and the ART 1_m algorithms. For ρ changing from 0.0 to 0.7 in steps of 0.1, each sub-figure in Fig. 6 depicts the resulting average distance for different values of L while sweeping α between 1.01 and 5.0 .

It seems natural to expect that, for a given value of ρ and a given value of the original ART 1 parameter L, there is an optimal value for the ART 1_m parameter α that will minimize the difference in behavior between the two algorithms. To find this relation between L and α for each ρ, we computed (for a given ρ and L) the value of α that minimizes the average distance between the learned patterns sets generated by the two algorithms. The results of these computations are shown in Fig. 7 [13]. Fig. 7(a) shows a family of curves (one for each value of ρ), that shows the optimal value of α as a function of L. Fig. 7(b) shows the resulting minimum average distance between learned sets for the same family of curves. As shown in Fig. 7(a), the optimum fit between parameters α and L is very slightly dependent on the value of ρ.

As can be concluded from Fig. 5, Fig. 6, Fig. 7, and the discussion in this Section, the behavior of the two algorithms is qualitatively the same although some slight quantitative differences can be observed. ART 1_m parameter α has a wider tuning range than original ART 1 parameter L. On the other hand, ART 1_m needs a slightly higher number of learning trials than the original ART 1. Also, there is an optimal adjustment between parameters α and L that minimizes the difference in behavior between the two algorithms, and this adjustment appears approximately independent of ρ.

13. Note that high values of ρ and L were omitted in this analysis, since in these cases the behavior of the two algorithms tends to be similar, regardless of the fit between parameters L and α.

V. Extending the ART 1_m Model to *Type-2* and *Type-1* Descriptions

The great advantage of the ART 1_m algorithm is its ability to produce a very simple *Type-3* hardware implementation, requiring only a binary valued memory template and only addition, subtraction and comparison operations, as well as a Winner-Take-All competition. Although *Type-2* and *Type-1* descriptions can be found that lead to the *Type-3* behavior of the ART 1_m algorithm described in this paper, these descriptions do not possess the hardware-attractive features of the *Type-3* implementation. Nevertheless, brief *Type-2* and a *Type-1* descriptions for this ART 1_m algorithm are presented in this Section.

A. A Type-2 ART 1_m Implementation

The change in weights must be smooth in a *Type-2* description. Every time an input pattern $\mathbf{I}$ is presented and an F_2 category node is selected for LTM storage, only a partial change in LTM traces is allowed. In this case, it is obvious that we can no longer use a binary valued weight template.

As seen in Section II, Fig. 2(c) shows the flow diagram of a *Type-3* implementation of the ART 1_m algorithm. Extending this diagram to a *Type-2* description is straightforward. The only box that needs to be changed is that corresponding to the update of weights. Instead of using the algebraic formula $\mathbf{z}_J(new) = \mathbf{I} \cap \mathbf{z}_J(old)$ we have to use a time domain differential equation that would lead to the same steady state. The following set of differential equations fulfills this requirement,

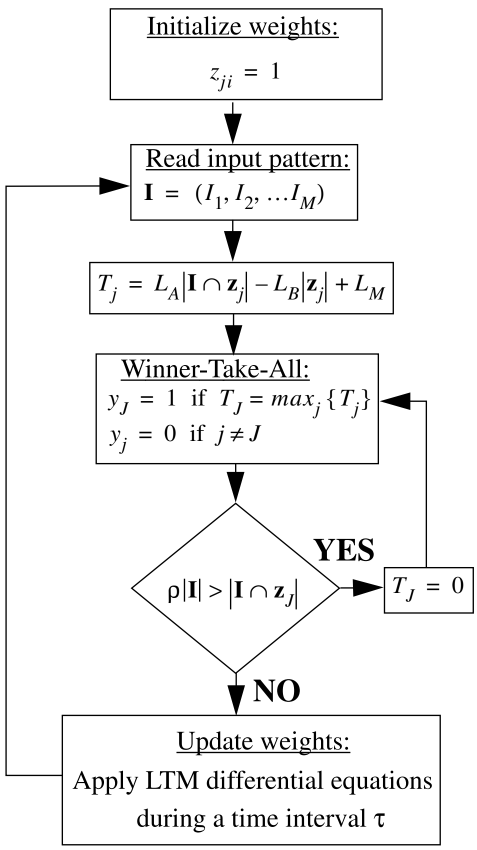

Fig. 8: ART 1_m algorithm *Type-2* implementation

$$\dot{z}_{ij} = K y_j [-z_{ij} + h(x_i)] \quad , \tag{65}$$

where K is a positive constant, $h(\cdot)$ a sigmoidal function, and x_i an STM variable given by,

$$x_i = I_i \sum_j y_j z_{ij} = I_i z_{iJ} \ . \tag{66}$$

If T_∞ is the time required for the LTM eqs. (65) to settle to their steady state, the update of weights (i.e., the simulation of eqs. (65)) would be allowed only for a time interval $\tau \ll T_\infty$ for each input pattern **I** presentation. As τ approaches T_∞, application of eqs. (65) or the update weights equation of Fig. 2(c) would become equivalent. Fig. 8 shows the flow diagram corresponding to this *Type-2* implementation of the ART 1_m algorithm.

B. A Type-1 ART 1_m Implementation

For a *Type-1* implementation, an appropriate set of STM equations must be found that leads to the flow diagram of Fig. 8 when the STM time constants are very small compared to the LTM ones. The following time domain STM differential equations would serve our purpose,

$$\begin{aligned} F_1: \qquad & \varepsilon \dot{x}_i = -x_i + (1 - A_1 x_i) J_i^+ - (B_1 + C_1 x_i) J_i^- \\ F_2: \qquad & \varepsilon \dot{x}_j = -x_j + (1 - A_2 x_j) J_j^+ - (B_2 + C_2 x_j) J_j^- \end{aligned} \tag{67}$$

where,

$$\begin{aligned} J_i^+ &= I_i + D_1 \sum_j f(x_j) z_{ij} \quad , \\ J_i^- &= \sum_j f(x_j) \quad , \\ J_j^+ &= g(x_j) + T_j \quad , \\ J_j^- &= \sum_{k \neq j} g(x_k) \quad . \end{aligned} \tag{68}$$

Parameters ε, A_1, B_1, C_1, A_2, B_2, C_2, and D_1 are positive and constant. Functions $f(\cdot)$ and $g(\cdot)$ are sigmoidal. Note that $y_j = f(x_j)$. Functions $g(\cdot)$ will be responsible for the resulting Winner-Take-All action of the F_2 layer. These STM equations are identical to those of the original ART 1 algorithm [Carpenter, 1987a], except that we use one weight template instead of two. However, the main difference lies in the way the terms T_j are computed. In this case T_j will be given by the following equation,

$$T_j = D_2 \left[L_A \sum_i h(x_i) z_{ij} - L_B \sum_i z_{ij} + L_M \right] \ . \tag{69}$$

where D_2 is constant and positive. Using eqs. (67)-(69) together with an STM *Reset System* will assure that if the STM time constants are very small compared to the LTM ones, the *Type-2* description of Fig. 8 results. The *Reset System* can be identical to that used in the original ART 1 system: each active input ($I_i = 1$) sends an excitatory signal of size P to an orienting subsystem A. Each F_1 node x_i which exceeds zero generates an inhibitory signal of size Q and sends it to A. The orienting subsystem A generates a nonspecific reset wave to F_2 whenever

$$\frac{|\mathbf{X}|}{|\mathbf{I}|} < \rho = \frac{P}{Q} , \tag{70}$$

where **I** is the input pattern and $|\mathbf{X}|$ is the number of F_1 nodes such that $x_i > 0$. The nonspecific reset wave shuts off active F_2 nodes until the input pattern **I** shuts off.

VI. Conclusions

This paper has presented, analyzed, and studied a modification to the original ART 1 algorithm. Such modification has drastic consequences from a hardware implementation point of view, in the sense that it extraordinarily simplifies the hardware requirements and components of the overall system and provides a very important increased performance potential. Although the modification produces some changes in the original behavior of the system, we have shown that all the computational properties of the original ART 1 algorithm are preserved. We have also performed exhaustive simulations to highlight the differences in behavior introduced by the modified system. Finally, we have sketched how to extend conceptually such a modified system to a non-*Fast Learning* description although this would lead to the loss of important hardware advantages.

We have used this ART 1_m model to implement a high performance, analog current mode, real-time clustering chip in a standard low cost 1.5μm CMOS process [Serrano, 1994, 1996]. Although we have used a specific circuit design technique (analog current mode), the ART 1 model described in this paper can be used with other circuit techniques. The only functions needed are binary storage, sums and/or subtractions, comparisons, and a Winner-Take-All action. The advantages of the ART 1_m model can be exploited using any hardware technique. We hope that the modifications introduced in this paper can be used by other neural hardware engineers regardless of the circuit design technique they choose to use.

VII. References

K. Bult and H. Wallinga (1987), "A Class of Analog CMOS Circuits Based on the Square-Law Characteristic of an MOS Transistor in Saturation," *IEEE Journal of Solid-State Circuits,* vol. SC-22, No. 3, pp. 357-365, 1987.

G. A. Carpenter and S. Grossberg (1987a), "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54-115, 1987.

G. A. Carpenter and S. Grossberg (1987b), "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, vol. 26, No. 23, pp. 4919-4930, 1 December 1987.

G. A. Carpenter and S. Grossberg (1990), "ART 3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures," *Neural Networks*, vol. 3, pp. 129-152, 1990.

G. A. Carpenter, S. Grossberg, and D. B. Rosen (1991a), "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks*, vol. 4, pp. 759-771, 1991.

G. A. Carpenter, S. Grossberg, and J. H. Reynolds (1991b), "ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network," *Neural Networks*, vol. 4, pp. 565-588, 1991.

G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen (1992), "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Transactions on Neural Networks*, vol. 3, No. 5, pp. 698-712, September 1992.

G. A. Carpenter and M. N. Gjaja (1994), “Fuzzy ART Choice Functions,” *Proceedings of the 1994 World Congress on Neural Networks (WCNN’94),* vol. I, pp. 713-722.

J. J. F. Cavanagh, *Digital Computer Arithmetic*, McGraw-Hill, 1985.

B. Gilbert (1990), “Current-Mode Circuits From a Translinear Viewpoint: A Tutorial,” in *Analogue IC Design: The Current-Mode Approach*, C. Toumazou, F. J. Lidgey, and D. G. Haigh (Eds.), IEE Circuits and Systems Series 2, Peter Peregrinus Ltd., London, UK, 1990, Chapter 2, pp. 11-91.

C. S. Ho, J. J. Liou, M. Georgiopoulos, G. L. Heileman, and C. Christodoulou (1994), “Analogue Circuit Design and Implementation of an Adaptive Resonance Theory (ART) Neural Network Architecture,” *International Journal of Electronics,* vol. 76, No. 2, pp. 271-291, 1994.

E. Sánchez-Sinencio, J. Ramírez-Angulo, B. Linares-Barranco, and A. Rodríguez-Vázquez (1989), “Operational Transconductance Amplifier-Based Nonlinear Function Synthesis,” *IEEE Journal of Solid-State Circuits,* vol. 24, No. 6, pp. 1576-1586, 1989.

T. Serrano-Gotarredona, B. Linares-Barranco, and J. L. Huertas (1994), “A CMOS VLSI Analog Current-Mode High-Speed ART 1 Chip,” *Proceedings of the 1994 IEEE International Conference on Neural Networks (ICNN’94),* Orlando, Florida, vol. 3, pp. 1912-1916.

T. Serrano-Gotarredona and B. Linares-Barranco (1996), “A Real-Time Clustering Microchip Neural Engine,” *IEEE Transactions on VLSI Systems,* accepted for publication.

D. H. Sheingold (1976), *Nonlinear Circuits Handbook,* Analog Devices Inc., Norwood, Massachusetts, U. S. A., 1976.

S. W. Tsay and R. W. Newcomb (1991), “VLSI Implementation of ART 1 Memories,” *IEEE Transactions on Neural Networks*, vol. 2, No. 2, pp. 214-221, March 1991.

D. C. Wunsch II, T. P. Caudell, C. D. Capps, R. J. Marks II, and R. A. Falk (1993), “An Optoelectronic Implementation of the Adaptive Resonance Neural Network,” *IEEE Transactions on Neural Networks*, vol. 4, No. 4, pp. 673-684, July 1993.

Appendix

During the writing of this paper other alternatives to the computation of the terms T_j of eq. (7) have been proposed [Carpenter, 1994] for a Fuzzy-ART architecture. Since ART 1 reduces to a particular case of Fuzzy-ART when the input pattern **I** is binary valued, any valid way of computing T_j in Fuzzy-ART should, in principle, be valid for ART 1 as well. The different T_j functions (also called ‘*distances*’ or ‘*choice functions*’) proposed in [Carpenter, 1994] when particularized for ART 1 result in the following formulations:

$$\begin{array}{ll} \text{Function 1:} & |\mathbf{I} \cap \mathbf{z}_j| - |\mathbf{z}_j| + \varepsilon\,(|\mathbf{z}_j| - |\mathbf{I} \cup \mathbf{z}_j|)\,, \\ \text{Function 2:} & |\mathbf{I} \cap \mathbf{z}_j| - |\mathbf{z}_j| + \varepsilon\,(|\mathbf{z}_j| - |\mathbf{I}|)\,. \end{array} \tag{71}$$

Note that these functions are also based on the subtraction operation, as in ART 1_m, but are computationally more expensive since either $|\mathbf{I} \cup \mathbf{z}_j|$ or $|\mathbf{I}|$ has to be computed as well. The *choice function* that we have used in this paper would be equivalent to the following,

$$T_j = |\mathbf{I} \cap \mathbf{z}_j| - |\mathbf{z}_j| + \varepsilon|\mathbf{z}_j| = |\mathbf{I} \cap \mathbf{z}_j| - (1-\varepsilon)\,|\mathbf{z}_j|\ , \tag{72}$$

and parameter $\alpha = L_A/L_B > 1$ would have been equivalent to

$$\alpha = \frac{1}{1-\varepsilon}\ . \tag{73}$$

If all the original ART 1 properties are to be preserved, we know now that α has to be greater than one. This implies,

$$\alpha > 1 \quad \Leftrightarrow \quad 1 > \varepsilon > 0\ . \tag{74}$$

With respect to the *choice functions* in eq. (71), Function 2 is mathematically equivalent to eq. (72), because the only difference between the two is the term $-\varepsilon|\mathbf{I}|$. Since the input is common to all of the category nodes and does not change during a single presentation, this term effectively acts as a uniform negative bias on all of the category nodes, regardless of the pattern coded in their templates. Eq. (72), therefore, is more efficient because the input size computation is unnecessary.

Function 1 of eq. (71) is another valid *choice function*, but is also computationally more expensive than eq. (72). It can be shown that the original ART 1 computational properties are preserved when this function is used (provided $\varepsilon > 0$). To see this, substitute the equations of Section III whose numbers appear in the first column of Table 1 by the equations in the second column, and note that

$$\begin{aligned} \left|\mathbf{I} \cup \mathbf{z}_j\right| &\geq \left|\mathbf{z}_j\right|, |\mathbf{I}| \\ \left|\mathbf{I} \cap \mathbf{z}_j\right| &\leq \left|\mathbf{z}_j\right|, |\mathbf{I}| \\ \left|\mathbf{I} \cup \mathbf{z}_J\right| &= |\mathbf{I}| + \left|\mathbf{z}_J\right| - \left|\mathbf{I} \cap \mathbf{z}_J\right| \end{aligned} \tag{75}$$

are always satified (if we know that $\mathbf{I} \neq \mathbf{z}_j$ then the '$\geq$' and '$\leq$' signs in eq. (75) can be substituted by '>' and '<', respectively). Table 1 only provides the demonstrations for properties *A, B, E, G,* and *I* of Section III. Properties *C, D,* and *F* are automatically satisfied since they do not depend on the explicit formulation of T_j. With respect to properties *H* (Search Order) it can be shown that all of them are fulfilled if eqs. (35), (37), and (38) are changed to

$$\frac{1}{1-\varepsilon} < \frac{M}{M-1} \ , \tag{76}$$

$$\left|\mathbf{z}_j\right| < \left|\mathbf{z}_J\right| \qquad \text{and} \qquad \frac{\left|\mathbf{I} \cup \mathbf{z}_j\right| - \left|\mathbf{z}_j\right|}{\left|\mathbf{z}_J\right| - \left|\mathbf{z}_j\right|} < \varepsilon \ \text{, and} \tag{77}$$

$$\left|\mathbf{I} \cap \mathbf{z}_j\right| - \varepsilon\left|\mathbf{I} \cup \mathbf{z}_j\right| - (1-\varepsilon)\left|\mathbf{z}_j\right| > T_J(\mathbf{I}, \text{t=0}) \ , \tag{78}$$

respectively.

original equation	new equation
(15)	$T_{j_1} = \lvert\mathbf{z}_{j_1}\rvert - \varepsilon\lvert\mathbf{z}_{j_1}\rvert - (1-\varepsilon)\lvert\mathbf{z}_{j_1}\rvert = 0$ $T_{j_2} = \lvert\mathbf{z}_{j_1}\rvert - \varepsilon\lvert\mathbf{z}_{j_2}\rvert - (1-\varepsilon)\lvert\mathbf{z}_{j_2}\rvert = \lvert\mathbf{z}_{j_1}\rvert - \lvert\mathbf{z}_{j_2}\rvert < 0$
(16)	$T_{j_1} = \lvert\mathbf{z}_{j_1}\rvert - \varepsilon\lvert\mathbf{z}_{j_2}\rvert - (1-\varepsilon)\lvert\mathbf{z}_{j_1}\rvert = \varepsilon(\lvert\mathbf{z}_{j_1}\rvert - \lvert\mathbf{z}_{j_2}\rvert) < 0 \quad \text{if} \quad \varepsilon > 0$ $T_{j_2} = \lvert\mathbf{z}_{j_2}\rvert - \varepsilon\lvert\mathbf{z}_{j_2}\rvert - (1-\varepsilon)\lvert\mathbf{z}_{j_2}\rvert = 0$
(18),(19)	$T_J = \lvert\mathbf{I}\rvert - \varepsilon\lvert\mathbf{I}\rvert - (1-\varepsilon)\lvert\mathbf{I}\rvert = 0$ $T_j = \lvert\mathbf{I} \cap \mathbf{z}_j\rvert - \lvert\mathbf{z}_j\rvert + \varepsilon(\lvert\mathbf{z}_j\rvert - \lvert\mathbf{I} \cup \mathbf{z}_j\rvert) < 0 \quad \text{if} \quad \varepsilon > 0$
(24)	$T_J(new) = \lvert\mathbf{I} \cap \mathbf{z}_J(new)\rvert - \varepsilon\lvert\mathbf{I} \cup \mathbf{z}_J(new)\rvert - (1-\varepsilon)\lvert\mathbf{z}_J(new)\rvert =$ $= \lvert\mathbf{I} \cap \mathbf{z}_J(old)\rvert - \varepsilon\lvert\mathbf{I} \cup [\mathbf{I} \cap \mathbf{z}_J(old)]\rvert - (1-\varepsilon)\lvert\mathbf{I} \cap \mathbf{z}_J(old)\rvert =$ $= \lvert\mathbf{I} \cap \mathbf{z}_J(old)\rvert - \varepsilon\lvert\mathbf{I}\rvert - (1-\varepsilon)[\lvert\mathbf{I}\rvert + \lvert\mathbf{z}_J(old)\rvert - \lvert\mathbf{I} \cup \mathbf{z}_J(old)\rvert] \geq$ $\geq \lvert\mathbf{I} \cap \mathbf{z}_J(old)\rvert - \varepsilon\lvert\mathbf{I} \cup \mathbf{z}_J(old)\rvert - (1-\varepsilon)\lvert\mathbf{z}_J(old)\rvert = T_J(old)$
(29)	$O_j = \varepsilon\lvert\mathbf{z}_j\rvert - \varepsilon\lvert\mathbf{I}\rvert < O_J = \varepsilon\lvert\mathbf{z}_J\rvert - \varepsilon\lvert\mathbf{I}\rvert \quad \Leftrightarrow \quad \lvert\mathbf{z}_j\rvert < \lvert\mathbf{z}_J\rvert \quad (\varepsilon > 0)$
(32)	$O_j = \varepsilon\lvert\mathbf{z}_j\rvert - \varepsilon\lvert\mathbf{I}\rvert < O_J = \lvert\mathbf{I} \cap \mathbf{z}_J\rvert - \varepsilon\lvert\mathbf{I} \cup \mathbf{z}_J\rvert - (1-\varepsilon)\lvert\mathbf{z}_J\rvert <$ $< \lvert\mathbf{z}_J\rvert - \varepsilon\lvert\mathbf{I}\rvert - (1-\varepsilon)\lvert\mathbf{z}_J\rvert = \varepsilon\lvert\mathbf{z}_J\rvert - \varepsilon\lvert\mathbf{I}\rvert \quad \Rightarrow \quad \lvert\mathbf{z}_j\rvert < \lvert\mathbf{z}_J\rvert \quad (\varepsilon > 0)$
(33)	$O_j = \lvert\mathbf{z}_j\rvert - \varepsilon\lvert\mathbf{I}\rvert - (1-\varepsilon)\lvert\mathbf{z}_j\rvert < \lvert\mathbf{I} \cap \mathbf{z}_J\rvert - \varepsilon\lvert\mathbf{I}\rvert - (1-\varepsilon)\lvert\mathbf{z}_J\rvert <$ $< \rho\lvert\mathbf{I}\rvert - \varepsilon\lvert\mathbf{I}\rvert - (1-\varepsilon)\lvert\mathbf{z}_j\rvert \quad \Rightarrow \quad \lvert\mathbf{z}_j\rvert < \rho\lvert\mathbf{I}\rvert$
(53)	$\lvert\mathbf{I} \cap \mathbf{z}_j\rvert - \varepsilon\lvert\mathbf{I} \cup \mathbf{z}_j\rvert - (1-\varepsilon)\lvert\mathbf{z}_j\rvert < \lvert\mathbf{I}\rvert - M$
(54)	$\varepsilon > \dfrac{\lvert\mathbf{I} \cap \mathbf{z}_j\rvert + M - \lvert\mathbf{I}\rvert - \lvert\mathbf{z}_j\rvert}{\lvert\mathbf{I} \cup \mathbf{z}_j\rvert - \lvert\mathbf{z}_j\rvert}$

Table 1