CONTRIBUTED ARTICLE

A Modified ART 1 Algorithm more Suitable for VLSI Implementations

TERESA SERRANO-GOTARREDONA AND BERNABÉ LINARES-BARRANCO

National Microelectronics Centre, Sevilla, Spain

(Received 6 December 1994; accepted 13 November 1995)

Abstract—This paper presents a modification to the original ART 1 algorithm (Carpenter & Grossberg, 1987a, A massively parallel architecture for a self-organizing neural pattern recognition machine, Computer Vision, Graphics, and Image Processing, 37, 54–115) 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 1m) is the result of hardware motivated simplifications investigated during the design of an actual ART 1 chip [Serrano-Gotarredona et al., 1994, Proc. 1994 IEEE Int. Conf. Neural Networks (Vol. 3, pp. 1912–1916); Serrano-Gotarredona & Linares-Barranco, 1996, IEEE Trans. VLSI Systems, (in press)]. 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.

Copyright © 1996 Elsevier Science Ltd.

Keywords—Adaptive resonance theory (ART), VLSI-friendly algorithms, Mathematical and computational analysis.

1. INTRODUCTION

In 1987 Carpenter and Grossberg published the ART 1 algorithm in a brilliant and well-founded paper (Carpenter & Grossberg, 1987a), the first of a series of Adaptive Resonance Theory (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 & Grossberg, 1987b) and Fuzzy-ART (Carpenter et al., 1991b) architectures do the same but for analog input patterns. ART 3 (Carpenter & Grossberg, 1990) introduces a search process for ART architectures that can robustly cope with sequences of asynchronous analog input patterns in real time. ARTMAP (Carpenter et al., 1991a) and Fuzzy-ARTMAP (Carpenter et al., 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 < ρ ≤ 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.

• 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 & Grossberg, 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 (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. (1994) proposed a circuit technique for a Type-1 implementation; Tsay and Newcomb (1991) proposed a CMOS circuit technique that would realize a partial Type-2 implementation; Wunsch et al. (1993) have built an optical-based Type-3 implementation; elsewhere (Serrano-Gotarredona et al., 1994; Serrano-Gotarredona & Linares-Barranco, 1996) we present a CMOS VLSI Type-3 circuit.

This paper presents a modification to the original ART 1 algorithm (Carpenter & Grossberg, 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-Gotarredona et al., 1994; Serrano-Gotarredona & Linares-Barranco, 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 2 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 3 shows that all computational properties present in the original ART 1 architecture are preserved in the modified version. Section 4 studies the differences in behavior between the two descriptions and provides simulation results, and
Section 5 indicates how to extend the ART \( l_m \) Type-3 description to Type-2 and Type-1 models.

2. FROM THE ORIGINAL ART I 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 Figure 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 \( I \equiv (I_1, I_2, \ldots, I_M) \). The \( F_2 \) layer category that will become active is that which most closely represents the input pattern \( I \). If no pre-existing 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 \( T_j \):

\[
y_j = 1 \quad \text{if } T_j = \max_j \{ T_j \},
\]

\[
y_j = 0 \quad \text{otherwise.} \tag{2}
\]

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

\[
X_i = I_i \sum_j z_{ij}^{td} y_j. \tag{3}
\]

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:

\[
X_i = I_i z_{ij}^{td} \quad \text{or} \quad X = I \cap z_{ij}^{td}, \tag{4}
\]

where \( z_{ij}^{td} = (z_{ij}^{td_1}, z_{ij}^{td_2}, \ldots, z_{ij}^{td_M}) \). This top-down template will be compared with the original input pattern \( I \) according to a predetermined vigilance criterion, tuned by a vigilance parameter \( 0 < \rho \leq 1 \), so that two alternatives may occur:

(a) If \( \rho |I| \leq |I \cap z_{ij}^{td}| \), the active category \( J \) is accepted, and the system weights will be updated to incorporate this new knowledge.\(^4 \)

(b) If \( \rho |I| > |I \cap z_{ij}^{td}| \), the active category \( J \) is not valid for the given value of the vigilance parameter \( \rho \). 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:

\[
z_{ij}^{bu} (\text{new}) = \frac{L}{L - 1 + |z_{ij}^{td} (\text{old}) \cap I|} X_i
\]

\[
= \frac{L}{L - 1 + |z_{ij}^{td} (\text{old}) \cap I|} I_i z_{ij}^{td} (\text{old})
\]

\[
z_{ij}^{td} (\text{new}) = X_i = I_i z_{ij}^{td} (\text{old}) \tag{5}
\]

---

\(^1\) Bottom-up weights \( z_{ij}^{bu} \) may take any real value in the interval \([0, K]\), where \( K = L/(L - 1 + M) \), and \( L > 1 \) (Carpenter & Grossberg, 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 & Grossberg, 1987a).

\(^3\) In the fast learning (Type-3) model top-down weights \( z_{ij}^{td} \) may take only the values \("0" \) or \("1"\).

\(^4\) The notation \(|a|\) represents the cardinality of vector \( a \), i.e. \(|a| = \sum |a_i|\).
or, using vector notation:

\[
x^{\text{new}}_j = \frac{LI \cap x^{\text{old}}_j}{L - 1 + |I \cap x^{\text{old}}_j|},
\]

\[
x^M_j = I \cap x^M_j,
\]

where \( L \geq 1 \) 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 Figure 2a.

From a hardware implementation point of view, one of the first issues that come into consideration is that there are two templates of weights to be built. The set of bottom-up weights \( z_{ij}^{\text{bu}} \), each of which must store a real value belonging to the interval \([0, K] \), and the set of top-down weights \( z_{ij}^{\text{td}} \), 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 eqns (6) that the bottom-up set \( \{z_{ij}^{\text{bu}}\} \) and the top-down set \( \{z_{ij}^{\text{td}}\} \) contain the same information: each of these sets can be fully computed by knowing the other set. The bottom-set \( \{z_{ij}^{\text{bu}}\} \) is a normalized version of the top-down set \( \{z_{ij}^{\text{td}}\} \). 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 \( \{z_{ij}^{\text{bu}}\} \) and \( \{z_{ij}^{\text{td}}\} \) can be substituted by a single binary valued set \( \{z_{ij}\} \), and eqn (1) modified to take into account the normalization effect of the original bottom-up weights:

\[
T_j = L_A |I \cap z_j| - L_B |z_j| + L_M,
\]

where \( L_A \) and \( L_B \) are positive parameters that play the role of the original \( L \) and \( (L-1) \) parameters. 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 to ensure that \( T_j \geq 0 \) for all possible values of \( |I \cap z_j| \) and \( |z_j| \).

Replacing a division operation with a subtraction one is a very important hardware simplification with significant performance improvement potential. Figure 2c shows the final Type-3 ART \( I_m \) algorithm, the object of this paper. In the next sections, we will try to show that the price paid for this drastic

\[\text{footnote}^5\]

This type of modification is employed in the Fuzzy-ART model (Carpenter et al., 1991b), 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.

\[\text{footnote}^6\]

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.

\[\text{footnote}^7\]

In reality, parameter \( L_M \) has been introduced for hardware reasons (Serrano-Gotarredona et al., 1994; Serrano-Gotarredona & Linares-Barranco, 1996). In a software ART \( I_m \) implementation parameter \( L_M \) can be ignored.
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 in hardware constrains 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.

FIGURE 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) VLSI-friendly ART 1m.

FIGURE 3. Illustration of simplification process of the division operation: (a) original division operation; (b) piecewise linear approximation; (c) linear approximation.
(Cavanagh, 1985). Sequential techniques use simpler hardware but are slower, while a dedicated circuit is very large compared with 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:

$$T_j = \frac{\|I \cap z_j\|}{L - 1 + |z_j|},$$

(9)

$q$ addition/subtractions 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, the numerator and denominator in eqn (9) should be represented by 10-bit words. If, for a given input $I$, we want to differentiate between two terms $T_{j_1}$ and $T_{j_2}$ whose respective templates $z_{j_1}$ and $z_{j_2}$ differ in one bit, the $F_2$ layer (WTA) would need to resolve:

$$|\Delta T_{j_1,j_2}|_{\text{min}} = \left| \frac{L|I \cap z_{j_1}|}{L - 1 + |z_{j_1}|} - \frac{L|I \cap z_{j_2}|}{L - 1 + |z_{j_2}|} \right|_{\text{min}}.$$  \quad (10)

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

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

A reasonable minimum value for $L$ is 1.01. Therefore, if $M = 1000$ then $|\Delta T_{j_1,j_2}|_{\text{min}} \approx 10^{-8}$. On the other hand, it is easy to see that $|\Delta T_{j_1,j_2}|_{\text{max}}$ is close to but less than 1. Consequently, for each $T_j$ a dynamic range of:

$$\frac{|T_j|_{\text{max}}}{|T_j|_{\text{min}}} = 10^8$$  \quad (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 1m 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 (Sheingold, 1976; Bult & Wallinga, 1987; Sánchez-Sinencio et al., 1989; Gilbert, 1990), 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 division 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 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.

### 3. ON THE COMPUTATIONAL EQUIVALENCE OF THE ORIGINAL AND THE MODIFIED MODELS

Throughout the original ART 1 paper (Carpenter & Grossberg, 1987a), the authors 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 (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 1m, 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 & Grossberg, 1987a). In the remainder of this section we will show that these properties remain in the ART 1m architecture.

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

(a) **Direct access:** an input pattern $I$ 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 F2 layer and is accepted by the vigilance subsystem, so that no reset occurs.

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

\[
\begin{align*}
  z_{ij} = 0 & \Rightarrow I_i = 0 \quad \forall i = 1, \ldots, M, \\
  I_i = 1 & \Rightarrow z_{ij} = 1 \quad \forall i = 1, \ldots, M, 
\end{align*}
\]  

(13)

and there are some values of \( i \) such that \( I_i = 0 \) and \( z_{ij} = 1 \).

(c) Superset template: an input pattern I is said to be a superset template of a learned category \( z_j \) if \( z_j \subset I \).

(d) Mixed template: \( z_j \) and I are said to be mixed templates if neither \( I \subset z_j \) nor \( z_j \subset I \) are satisfied, and \( I \neq z_j \).

(e) Uncommitted node: an F2 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 F2 layer. Each category \( y_j \) is characterized by the set of weights that connect node \( y_j \) in the F2 layer to all nodes in the F1 layer, i.e., \( z_j \equiv (z_{1j}, z_{2j}, \ldots, z_{Mj}) \). Suppose that two of these categories, \( y_{j1} \) and \( y_{j2} \), are such that \( z_{j1} \subset z_{j2} \) (\( z_{j1} \) is a subset template of \( z_{j2} \)). Now consider two input patterns \( I^{(1)} \) and \( I^{(2)} \) such that:

\[
\begin{align*}
  I^{(1)} &= z_{j1} = (z_{1j1}, z_{2j1}, \ldots, z_{Mj1}), \\
  I^{(2)} &= z_{j2} = (z_{1j2}, z_{2j2}, \ldots, z_{Mj2}).
\end{align*}
\]  

(14)

The direct access to subset and superset property assures that input \( I^{(1)} \) will have direct access to category \( y_{j1} \), and that input \( I^{(2)} \) will have direct access to category \( y_{j2} \).

Proof.

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

\[
\begin{align*}
  T_{j1} &= L_A \sum_{i=1}^{M} I_{i}^{(1)} z_{ih} - L_B |z_{ih}| + L_M \\
  &= L_A |I^{(1)}| - L_B |I^{(1)}| + L_M, \\
  T_{j2} &= L_A \sum_{i=1}^{M} I_{i}^{(2)} z_{ih} - L_B |z_{ih}| + L_M \\
  &= L_A |I^{(2)}| - L_B |I^{(2)}| + L_M.
\end{align*}
\]  

(15)

Since \(|I^{(1)}| < |I^{(2)}|\), it follows that (remember \( L_B > 0 \)) \( T_{j1} > T_{j2} \). If pattern \( I^{(2)} \) is presented at the input layer of the network, it would be:

\[
\begin{align*}
  T_{j1} &= L_A \sum_{i=1}^{M} I_{i}^{(2)} z_{ih} - L_B |z_{ih}| + L_M \\
  &= L_A |I^{(1)}| - L_B |I^{(1)}| + L_M, \\
  T_{j2} &= L_A \sum_{i=1}^{M} I_{i}^{(1)} z_{ih} - L_B |z_{ih}| + L_M \\
  &= L_A |I^{(2)}| - L_B |I^{(2)}| + L_M. 
\end{align*}
\]  

(16)

In order to guarantee that \( T_{j1} > T_{j2} \), the condition:

\[
L_A > L_B
\]  

(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 \( I \) has direct access to a node \( y_j \) which has perfectly learned \( I \).

Proof.

In the case of the ART 1m algorithm, in order to prove that \( I \) has direct access to \( y_j \) we need to show that: (i) \( y_j \) is the first F2 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 \( z_j = I \), we need to prove:

\[
T_j = L_A |I| - L_B |I| > L_A |I \cap z_j| - L_B |z_j| = T_j.
\]  

(18)

Suppose first that \(|z_j| > |I|\). Since \(|I| \geq |I \cap z_j|\) is always true, then eqn (18) is satisfied:

\[
T_j = L_A |I| - L_B |I| \geq L_A |I \cap z_j| - L_B |I| > L_A |I \cap z_j| - L_B |z_j| = T_j.
\]  

(19)

Suppose that \(|z_j| \leq |I|\). Then, since \( z_j \neq I \), it follows that \(|I| > |I \cap z_j|\). Finally, since \(|z_j| \geq |I \cap z_j|\) is always true, it follows that:

\[
T_j = L_A |I| - L_B |I| > L_A |I \cap z_j| - L_B |I \cap z_j| \geq L_A |I \cap z_j| - L_B |z_j| = T_j.
\]  

(20)

\(^8\) In the original ART 1 paper it was 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 F2 node results in a change of \( T_j \) terms through the influence of the top-down connections.
Property (ii) is directly satisfied because:

$$|I \cap y_j| = |I| \geq \rho |I|, \forall \rho \in [0, 1].$$

(21)

Finally, property (iii) also holds, because after node $y_j$ is selected as the winning category, its weight template $z_j$ will remain unchanged [because $z_j(new) = I \cap z_j(old) = I = z_j(old)$], and consequently the inputs to the $F_2$ layer $T_j$ will remain unchanged.

\[\square\]

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

Whenever an input pattern $I$ is presented for the first time to the ART 1 system, a set of $\{T_j\}$ values is formed to 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:

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 $I$) may be altered through the top-down "feed-back" connections. In a Type-3 description (see Figure 2) the initial terms $T_j$ remain unchanged, independently of what happens in the $F_2$ layer. Therefore, this theorem is implicitly 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 & Grossberg, 1987a):

The Order Function ($O_j > O_1 > O_2 > \ldots$) 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 1m (both Type-3) implementations.\(^9\) If $T_j$ is reset by the vigilance subsystem, the values of $T_j, T_{j+1}, \ldots$ will not change. Therefore, the new order sequence is $O_j > O_i > \ldots$ and the original second largest value $O_h$ will be selected as the winner. If $T_h$ is now set to zero, $O_h$ 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 & Grossberg, 1987a).

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

This theorem (page 93 of Carpenter & Grossberg, 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 $I \cap z_j$ remains constant during learning, so that learning does not trigger reset of $y_j$ by the vigilance subsystem.

\begin{proof}
In this case, if $y_j$ is the winning category accepted by the vigilance subsystem, from eqn (8) we obtain:

$$T_j = L_a |I \cap z_j| - L_b |z_j| + L_m.$$ 

(22)

The update rule is:

$$z_j(new) = I \cap z_j(old),$$

(23)

and the new $T_j$ value is given by:

$$T_j(new) = L_a |I \cap z_j(new)| - L_b |z_j(new)| + L_m$$

$$= L_a |I \cap z_j(old)| - L_b |z_j(old)| + L_m$$

$$\geq L_a |I \cap z_j(old)| - L_b |z_j(old)| + L_m$$

$$= T_j(old).$$

(24)

Therefore, learning confirms the choice of $y_j$, and by eqn (23) the set $I \cap z_j$ remains constant.
\[\square\]

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 1m system. Each template set $z_j \equiv (z_{1j}, z_{2j}, \ldots, z_{Mj})$ is updated every time category $y_j$ is selected by the winner-take-

\(^9\) 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 & Grossberg, 1987a). This is because the demonstration is performed for a Type-1 description of ART 1.
all F<sub>2</sub> layer and accepted by the vigilance subsystem. Sometimes template \( z_j \) may be changed, and at others it may remain unchanged. Let us call the times \( t_j \) suffers a change \( t_j^{(1)} < t_j^{(2)} < \ldots < t_j^{(i)} \). Since the vector (or template) \( z_j \) of a committed node has \( M \) components (of which, at the most, \( M - 1 \) are set to '1'), and by eqn (23) each component can only change from '1' to '0' but not from '0' to '1', it follows that template \( z_j \) can, at the most, suffer \( M - 1 \) changes:

\[ r_j \leq M - 1. \quad (25) \]

Since template \( z_j \) will remain unchanged after time \( t_j^{(i)} \), it is concluded that the complete LTM memory will suffer no change after time:

\[ t_{\text{learn}} = \max \{ t_j^{(i)} \}. \quad (26) \]

If there is a finite number of nodes in the F<sub>2</sub> layer \( t_{\text{learn}} \) 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<sub>m</sub> architectures. Therefore, the following theorem (page 95 of Carpenter & Grossberg, 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 \( z_j \) remains constant except for at most \( M - 1 \) times \( t_j^{(1)} < t_j^{(2)} < \ldots < t_j^{(i)} \) at which it progressively loses elements, leading to the

**Subset Recoding Property:**

\[ z_j(t_j^{(i)}) \supset z_j(t_j^{(i-1)}) \supset \ldots \supset z_j(t_j^{(1)}). \quad (27) \]

The LTM traces \( z_j(t) \) such that \( i \notin z_j(t_j^{(i)}) \) decrease to zero. The LTM traces \( z_j(t) \) such that \( i \in z_j(t_j^{(i)}) \) remain always at '1'. The LTM traces such that \( i \in z_j(t_j^{(i)}) \) but \( i \notin z_j(t_j^{(i)}) \) stay at '1' for times \( t < t_j^{(i)} \) but will change to and stay at '0' for times \( t \geq t_j^{(i)} \).

**G. Direct Access after Learning Stabilizes (Theorem 6 of Original ART 1)**

Assuming F<sub>2</sub> has a finite number of nodes, the present theorem (page 98 of Carpenter & Grossberg, 1987a) states the following:

After recognition learning has stabilized in response to an arbitrary list of binary input patterns, each input pattern I either has direct access to the node \( y_j \) which possesses the largest subset template with respect to I, or I cannot be coded by any node of F<sub>2</sub>. In the latter case, F<sub>2</sub> contains no uncommitted nodes.

**Proof**

Since learning has already stabilized I can be coded only by a node \( y_j \) whose template \( z_j \) is a subset template with respect to I. Otherwise, once \( y_j \) becomes active, the set \( z_j \) would contract to \( z_j \cap I \), thereby contradicting the hypothesis that learning has already stabilized. Thus, if 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<sub>2</sub> node activated by I. We need to show that if \( y_j \) 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:

\[ |I \cap z_j| = |z_j| < \rho |I|. \quad (28) \]

If \( y_J \) and \( y_J \) are nodes with subset templates with respect to I, then:

\[ O_J = L_A |z_j| - L_B |z_j| + L_M < O_J = L_A |x_I| - L_B |x_J| + L_M. \quad (29) \]

Since \( (L_A - L_B) |z_j| \) is an increasing function of \( |z_j| \):

\[ |z_j| < |x_I|. \quad (30) \]

and

\[ R_J = \frac{|I \cap z_j|}{|I|} = \frac{|z_j|}{|I|} < R_J = \frac{|I \cap z_j|}{|I|} = \frac{|z_j|}{|I|}. \quad (31) \]

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 I \(|I \cap z_J| < |z_J|\), but another node \( y_J \) with a subset template is activated in the course of search. We need to show that \(|I \cap z_J| = |z_J| < \rho |I|\), so that \( y_J \) is reset. We know that:

\[ O_J = (L_A - L_B) |z_J| + L_M < O_J = L_A |I \cap z_J| - L_B |z_J| + L_M < (L_A - L_B) |z_J| + L_M, \quad (32) \]

which implies that \(|z_J| < |z_J|\). Since \( y_J \) cannot be chosen, it must be reset by the vigilance subsystem which means that \(|I \cap z_J| < \rho |I|\). Therefore:

\[ L_A |z_j| - L_B |z_j| < L_A |I \cap z_J| - L_B |z_J| < L_A \rho |I| - L_B |z_J|, \quad (33) \]

which implies that

\[ |I \cap z_J| = |z_J| < \rho |I|. \quad (34) \]

**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
l\(_n\) architecture. The modified theorem states the following:

Suppose:

\[
\frac{L_A}{L_n} < \frac{M}{M-1}, \tag{35}
\]

and that input pattern I satisfies:

\[
|I| \leq M - 1. \tag{36}
\]

Then F\(_{j}\) nodes are searched in the following order, if they are searched at all.

Subset templates with respect to 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 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 I. If all subset templates are searched and if both superset templates \(z_I\) and mixed templates \(z_j\) exist, then \(y_j\) will be searched before \(y_j\) if and only if:

\[
|z_I| < |z_j| \text{ and } \frac{|I| - |I \cap z_I|}{|z_I| - |z_I|} < \frac{L_A}{L_n}. \tag{37}
\]

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:

\[
L_A|I \cap z_I| - L_B|z_I| + L_M > T_2(I, t = 0), \tag{38}
\]

where

\[
T_2(I, t = 0) = L_A \sum z_I(0) - L_B \sum z_I(0) + L_M.
\]

The proof has several parts:

(a) First we show that a node \(y_j\) with a subset template \((I \cap z_I) = z_I\) is searched before any node \(y_j\) with a non-subset template. In this case:

\[
O_j = L_A|I \cap z_I| - L_B|z_I| + L_M = |I \cap z_I|
\]

\[
\times \left( L_A - L_B \frac{|z_I|}{|I \cap z_I|} \right) + L_M. \tag{39}
\]

Now, note that:

\[
\frac{|z_I|}{|I \cap z_I|} \geq \frac{M}{M-1} \tag{40}
\]

because\(^{11}\)

\[
\frac{|z_I|}{|I \cap z_I|} \geq \frac{M}{|z_I| - 1} = \frac{M - 1}{M - 2} > \frac{M}{M-1}. \tag{41}
\]

From eqns (35), (39) and (41), it follows that:

\[
O_j < |I \cap z_I|L_B \left( \frac{L_A}{L_B} - \frac{M}{M-1} \right) + L_M < L_M. \tag{42}
\]

On the other hand:

\[
O_j = (L_A - L_B)|z_I| + L_M > L_M. \tag{43}
\]

Therefore:

\[
O_j > O_j. \tag{44}
\]

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

Suppose two subset templates of I, \(z_I\) and \(z_j\) such that \(|z_I| > |z_j|\). Then:

\[
O_j = (L_A - L_B)|z_I| + L_M > (L_A - L_B)|z_j|
\]

\[
+ L_M = O_j. \tag{45}
\]

Therefore, node \(y_j\) will be searched before node \(y_I\). By eqn (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 \(y_j\):

\[
O_j = L_A|I| - L_B M + L_M \leq L_A(M - 1) - L_B M + L_M
\]

\[
= L_B \left( \frac{L_A}{L_B} (M - 1) - M \right) + L_M
\]

\[
< L_B \left( \frac{M}{M - 1} (M - 1) - M \right) + L_M
\]

\[
= L_M < (L_A - L_B)|z_I| + L_M = O_j. \tag{46}
\]

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

(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:

\[
O_j = L_A|I| - L_B|z_I| + L_M > L_A|I| - L_B M
\]

\[
+ L_M = O_j. \tag{47}
\]

If there is more than one superset template, the one with the smallest \(|z_I|\) will be activated. Since \(|I \cap z_I| = |I| \geq |I|\), 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 eqn (37) holds:

\(^{11}\) We assume that \(y_j\) is not an uncommitted node \((|z_I| < M)\).
As the ratio \( L_A / L_B \) increases it is more likely that eqn (54) be satisfied, and hence uncommitted nodes are chosen before coded nodes, regardless of the vigilance parameter value \( \rho \).

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

\[
O_{1h} > O_1 > O_{11} > \ldots,
\]

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

4. 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 eqn (55). Assuming that both models, at a certain time, have identical weight templates \( \{ z_j \} \), and the same input pattern \( I \) is given, eqn (55) has the following two formulations:

Original ART 1:

\[
\frac{|I \cap z_{j_1}|}{L - 1 + |z_{j_1}|} > \frac{|I \cap z_{j_2}|}{L - 1 + |z_{j_2}|} > \ldots
\]

Modified ART 1:

\[
\frac{L_A}{L_B} \left| I \cap z_{l_1} \right| - |z_{l_1}| > \frac{L_A}{L_B} \left| I \cap z_{l_2} \right| - |z_{l_2}| > \ldots
\]

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 with all \( |z_j| \) terms, then the ordering depends exclusively on the values of \( |I \cap z_j| \),

\[
\left| I \cap z_{j_1} \right| > \left| I \cap z_{j_2} \right| > \left| I \cap z_{j_3} \right| > \ldots
\]

If \( L \) is very close to 1, then the ordering depends on the ratios:

\[
\frac{|I \cap z_{j_1}|}{|z_{j_1}|} > \frac{|I \cap z_{j_2}|}{|z_{j_2}|} > \frac{|I \cap z_{j_3}|}{|z_{j_3}|} > \ldots
\]
Likewise, for the ART $l_m$ description, the ordering is modulated by a single parameter $\alpha = L_A/L_B > 1$. If $\alpha$ is extremely large, the situation in eqn (57) results. However, for $\alpha$ very close to 1, the ordering depends on the differences:

$$|I \cap z_i| - |z_i| > |I \cap z_i| - |z_i| > |I \cap z_i| - |z_i| > \ldots$$

(59)

Obviously, the behavior of the two ART 1 descriptions will be identical for large values of $L$ and $\alpha$. However, moderate values of $L$ and $\alpha$ 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 $\rho$: if $\rho$ 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 $\rho$. Therefore, very similar behavior between ART 1 and ART $l_m$ will be expected for high values of $\rho$, while more differences in behavior might be apparent for smaller values of $\rho$.

In order to compare the two algorithms’ behavior, we have performed exhaustive simulations using randomly generated training patterns sets. As an illustration of a typical case where the two algorithms produce different learned templates, Figure 4 shows the evolution of the memory templates for both the ART 1 and the ART $l_m$ algorithms, using a randomly generated training set of 10 patterns with 25 pixels each. Weight templates for original ART 1 are named $z_j$, while for ART $l_m$ they are named $z'_j$. The vigilance parameter was set to $\rho = 0.4$. For the original ART 1 $L = 5$ and for the ART $l_m$ $\alpha = 2$. In Figure 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 corresponding $z_j$ template suffered modifications due to learning. If the box is drawn with a dashed line, learning did not alter the corresponding $z_j$ template. Both algorithms stabilized their weights in two training trials. Looking at the learned templates we can see that input patterns 4 and 5 clustered in the same category for both algorithms ($z_4$ for original ART 1 and $z'_4$ for ART $l_m$). This also occurred for patterns 6 and 8 ($z_3$ and $z'_3$) and for patterns 3, 9, and 10 ($z_5$ and $z'_5$). 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 $z_1$, while pattern 2 remained independent in category $z_2$. In the ART $l_m$ algorithm patterns 1 and 2 clustered together into category $z'_1$, while pattern 7 remained independent in category $z'_2$.

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

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

(60)

where

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

(61)

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

$$\sum_{i=1}^{Q} d(z_i, z'_i).$$

(62)

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

$$D = d(z_1, z'_1) + d(z_2, z'_2) + d(z_3, z'_3) + d(z_4, z'_4) + d(z_5, z'_5) = 7.$$ 

(63)

In general, we can define the distance between two patterns sets $A = \{a_j\}_{j=1}^{Q}$ and $B = \{b_j\}_{j=1}^{Q}$ as:

$$D(A, B) = \min_{(l_1, l_2, \ldots, l_Q)} \left[ \sum_{i=1}^{Q} d(a_i, b_{l_i}) \right].$$

(64)

In the case of Figure 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 $P_2$ 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 Figure 4 many times for different sets of randomly generated training patterns and sweeping the values of $\rho$, $L$, and $\alpha$. For each combination of $\rho$, $L$, and $\alpha$ values, we repeated the simulation 100 times for different training patterns sets, and computed the average.

---

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".
number of learned categories, learning trials, and distance between learned categories, as well as their corresponding standard deviations. Figures 5 and 6 present the results of these simulations. Figure 5a shows how the average number of learned categories changes with $L$ (from 1.01 to 40) for different values of $\rho$, for the original ART 1. As $\rho$ decreases, parameter $L$ has more control on the average number of learned categories. Figure 5b shows the standard deviation for the number of learned categories of Figure 5a. 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 $\rho$) and for large values of $\rho$ (independently of $L$). Figures 5c and 5d show the same as Figures 5a and 5b, respectively, for the ART 1$_m$ algorithm. As we can see, parameter $\alpha$ (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
FIGURE 5. Simulated results comparing behavior between ART 1 and ART 1_m.
hand, ART 1m presents a slightly higher standard deviation than the original ART 1. Nevertheless, the qualitative behavior of both algorithms is similar. Figures 5e and 5f show the average number of learning trials and their corresponding deviations, needed by the original ART 1 algorithm to stabilize its learned weights. Figures 5g and 5h show the same for the ART 1m algorithm. As we can see, the ART 1m algorithm needs a slightly higher average number of learning trials to stabilize. Also, the standard deviation observed for the ART 1m algorithm is slightly higher. Finally, Figure 6 shows the resulting average distances [as defined by eqn (64)] between learned categories of the ART 1 and the ART 1m algorithms. For ρ changing from 0.0 to 0.7 in steps of 0.1, each subfigure in Figure 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 1m 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 Figure 7.13 Figure 7a shows a family of curves (one for each value of ρ), that show the optimal value of α as a function of L. Figure 7b shows the resulting minimum average distance between learned sets for the same family of curves. As shown in Figure 7a, the optimum fit between parameters α and L is very slightly dependent on the value of ρ.

As can be concluded from Figures 5, 6 and 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 1m parameter α has a wider tuning range than original ART 1 parameter L. On the other hand, ART 1m 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.

5. EXTENDING THE ART 1m MODEL TO TYPE-2 AND TYPE-1 DESCRIPTIONS

The great advantage of the ART 1m 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 1m 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 1m algorithm are presented in this section.

(a) A Type-2 ART 1m Implementation

The change in weights must be smooth in a Type-2 description. Every time an input pattern I is presented and an F2 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 2, Figure 2c shows the flow diagram of a Type-3 implementation of the ART 1m algorithm. Extending this diagram to a Type-2 description is straightforward. The only box that needs to be changed is that corresponding to the

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.
FIGURE 7. Optimal parameters fit between ART 1 and ART $1_n$. 
update of weights. Instead of using the algebraic formula \( z_j(\text{new}) = I \cap z_j(\text{old}) \) 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:

\[
\dot{z}_j = Ky_j[-z_j + h(x_j)], \tag{65}
\]

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

\[
x_i = I_i \sum_j y_j z_{ij} = I_i z_{ij}. \tag{66}
\]

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

(b) A Type-1 ART \( I_m \) Implementation

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

\[
F_1 : \dot{e}_i = -x_i + (1 - A_1 x_i) J^+_i - (B_1 + C_1 x_i) J^-_i,
\]

\[
F_2 : \dot{e}_j = -x_j + (1 - A_2 x_j) J^+_j - (B_2 + C_2 x_j) J^-_j, \tag{67}
\]

where

\[
J^+_i = I_i + D_1 \sum_j f(x_j) z_{ij},
\]

\[
J^-_i = \sum_j f(x_j),
\]

\[
J^+_j = g(x_j) + T_j,
\]

\[
J^-_j = \sum_{k \neq j} g(x_k). \tag{68}
\]

Parameters \( e, A_1, B_1, C_1, A_2, B_2, C_2, \) 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 & Grossberg, 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:

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

where \( D_2 \) is constant and positive. Using eqns (67)–(69) together with an STM reset system will assure that if the STM time constants are very small compared with the LTM ones, the Type-2 description of Figure 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 \( x_j \) 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

\[
\frac{|X|}{|I|} < \rho = \frac{P}{Q}, \tag{70}
\]

where \( I \) is the input pattern and \(|X|\) 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.
6. CONCLUSIONS

This paper has presented, analyzed, and studied a modification to the original ART I 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 I 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 I model to implement a high performance, analog current mode, real-time clustering chip in a standard low cost 1.5 μm CMOS process (Serrano-Gotarredona et al., 1994; Serrano-Gotarredona & Linares-Barranco, 1996). Although we have used a specific circuit design technique (analog current mode), the ART I 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 I 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.

REFERENCES


APPENDIX

During the writing of this paper other alternatives to the computation of the terms $T_j$ of eqn (7) have been proposed (Carpenter & Gjaja, 1994) for a Fuzzy-ART architecture. Since ART I 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 I as well. The different $T_j$ functions (also called "distances" or "choice functions") proposed in Carpenter & Gjaja (1994) when particularized for ART I result in the following formulations:

$$\text{Function 1:} \quad ||I \cap z_j|| - ||z_j|| + c(||z_j|| - ||I \cup z_j||),$$

$$\text{Function 2:} \quad ||I \cap z_j|| - ||z_j|| + c(||z_j|| - ||I||). \quad (A.1)$$

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

$$T_j = ||I \cap z_j|| - ||z_j|| + c(||z_j|| - ||I \cup z_j||), \quad (A.2)$$

and parameter $\alpha = \frac{L_z}{L_z} > 1$ would have been equivalent to

$$\alpha = 1 - \frac{e}{1 - e}. \quad (A.3)$$
TABLE A.1

<table>
<thead>
<tr>
<th>Original Equation</th>
<th>New Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(15) $T_\alpha =</td>
<td>z_n</td>
</tr>
<tr>
<td>(16) $T_\lambda =</td>
<td>z_n</td>
</tr>
<tr>
<td>(18), (19) $T_j =</td>
<td>l</td>
</tr>
<tr>
<td>(24) $T_j (new) =</td>
<td>l \cap z_j</td>
</tr>
<tr>
<td>$=</td>
<td>l \cap z_j</td>
</tr>
<tr>
<td>(29) $O_j = \varepsilon</td>
<td>z_j</td>
</tr>
<tr>
<td>(32) $O_j = \varepsilon</td>
<td>z_j</td>
</tr>
<tr>
<td>$&lt;</td>
<td>z_j</td>
</tr>
<tr>
<td>(33) $O_j =</td>
<td>z_j</td>
</tr>
<tr>
<td>$&lt; \rho</td>
<td>l</td>
</tr>
<tr>
<td>(53) $</td>
<td>l \cap z_j</td>
</tr>
<tr>
<td>(54)</td>
<td></td>
</tr>
</tbody>
</table>

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

$$\alpha > 1 \Leftrightarrow 1 > \varepsilon > 0. \quad (A.4)$$

With respect to the choice functions in eqn (A.1), Function 2 is mathematically equivalent to eqn (A.2), because the only difference between the two is the term $-\varepsilon |l|$. 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. Equation (A.2), therefore, is more efficient because the input size computation is unnecessary.

Function 1 of eqn (A.1) is another valid choice function, but is also computationally more expensive than eqn (A.2). 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 3 whose numbers appear in the first column of Table A.1 by the equations in the second column, and note that:

$$\left| |l \cup z_j| - |l| \right| \geq |z_j|, |l|$$

$$\left| |l \cap z_j| \right| \leq |z_j|, |l|$$

$$|l \cup z_j| = |l| + |z_j| - |l \cap z_j|$$

are always satisfied (if we know that $l \neq z_j$, then the " $\geq$ " and " $\leq$ " signs in eqn (A.5) can be substituted by " $>$ " and " $<$ ", respectively). Table A.1 only provides the demonstrations for properties $A$, $B$, $E$, $G$, and $I$ of Section 3. Properties $C$, $D$, and $F$ are automatically satisfied since they do not depend on the explicit formulation of $T$. With respect to properties $H$ (search order) it can be shown that all of them are fulfilled if eqns (35), (37), and (38) are changed to:

$$\frac{1}{1 - \varepsilon} \leq \frac{M}{M - 1}, \quad (A.6)$$

$$|z_j| < |z_j| \text{ and } \frac{|l \cup z_j| - |z_j|}{|z_j| - |z_j|} < \varepsilon, \quad (A.7)$$

and

$$|l \cap z_j| - \varepsilon |l \cup z_j| - (1 - \varepsilon) |z_j| > T_j (l, t = 0), \quad (A.8)$$

respectively.