Systolic Hebb Agnostic Resonance Perceptron (SHARP): a Neural Network Model Inspired By the Topological Organization of the Cerebral Cortex, which Implements Virtual Parallelism on...

Systolic Hebb Agnostic Resonance Perceptron (SHARP): a Neural Network Model Inspired By the Topological Organization of the Cerebral Cortex, which Implements Virtual Parallelism on Von Neumann Computers

Luca Marchese

ORC-ID: http://orcid.org/0000-0001-7903-7540
NCAGE: AK845

INNS Senior Member, IEEE Member, IEEE-CIS Member, Italy

luca.marchese@synaptics.org

Abstract. This document describes the neural network model Systolic Hebb Agnostic Resonance Perceptron (SHARP) and explains the characteristics underpinning this connectionist paradigm inspired by the cerebral cortex. The proposed model is a classifier that can output an array of recognition strength values associated with specific categories. The concepts of sparse distributed storage (SDS) and sparse distributed code (SDC) are explained in a framework linking mini- and macro-column scale functionalities of the cerebral cortex. The learning process is executed in a single step. The pattern recognition speed on Von Neumann computers is independent of the number of learned prototypes and originates from the stimulus-driven synaptic addressing property (SDSA). This paper explains the mathematical model that emulates the neural network in a software implementation. The author demonstrates the efficiency of the model with comparative tests on large artificial databases, discusses why the model is tailored for deep hierarchical learning and describes how it can be used to execute, in real time, large neurocognitive networks on serial computers. The author wishes to note that the model is biologically inspired but is not proposed as biologically plausible. With this awareness, the author uses concepts such as mini-column and macro-column in order to organize the neural network topology.

Keywords: SHARP, SDC, SDS, SDSA, Neo-Cortex, Mini-Column, Macro-Column, Virtual Parallelism, Machine Learning, Rule extraction

I. Introduction

SHARP is a neural network inspired by the topological organization of the cerebral cortex and by a hypothesis regarding the functionalities of the mini-columns and macro-columns. The efficiency of its execution on serial computers is demonstrated with software simulations and pattern recognition tests on large databases. The neural network has been presented in [1] with analogic spiking neuron models and delayed synapses used as spike memories. The goal of this paper is to explain the most relevant characteristic of the network: the stimulus-driven synaptic addressing (SDSA) property. SDSA enables the input stimulus to directly address the small portion of the synapses and neurons required to perform the recognition process. Indeed, the recognition time on a serial computer is independent of the number of the learned patterns, while the learning activity is performed in a single step. Due to this feature, the title of the paper refers to this neural network using the words “virtual parallelism”.

There are different criteria used to classify neural networks, which are based on the architecture, the propagation of signals (i.e., feedforward vs. recurrent), the learning type (i.e., supervised vs. unsupervised) or simply the target behavior (e.g., associative memory, classifier, or function approximation). Other criteria and sub-criteria can be used to classify neural networks. Any model could be easily mapped on a spiking neural network using the firing rate codification. Other coding schemes, in which the timing or the presence of the single spike is meaningful (pulsed neural networks), require more complicated algorithms and/or modifications of the original model.

The model presented in this work is a feedforward supervised classifier that can be mapped on a pulsed neural network and can be implemented on software within an agile framework using only set theory (Fig. 1). The neural network model presented in this work is not compliant with the criteria defined by Maass [2] for third-generation neural networks. Indeed, the algorithm does not need to discretize the time in order to compute the evolution of the membrane potential of the neurons which is completed within the timeframe required for processing the entire macro-column. This time elapses from the firing of the first mini-column to the firing of the last mini-column. The spatiotemporal correlation management is delegated to the delays of the synapses, which are updated in the learning process.

To the best knowledge of the author, the neural network model presented in this paper is the first having the following characteristics:

- instantaneous learning;
- recognition time, in the software simulation on a serial computer, that is independent of the number of the learned patterns;

 ORC-ID: http://orcid.org/0000-0001-7903-7540 NCAGE: AK845

Email: luca.marchese@synaptics.org web: www.synaptics.org

- an architecture that can be mapped on a pulsed neural network (the specifications for a digital pulsed realization are not yet defined);
- simulation with basic math operations
- formal description with simple set theory; and
- rule extraction.

Although the analogic version presented in [1] uses biologically implausible neurons that combine a resonator and a perfect integrator [3], the network can be built by connecting the more biologically plausible RF (resonate and fire) [4] and LIF (leaky integrate and fire) neurons [5], provided that the decay of the membrane potential is correctly tuned (with a biologically implausible function).

To focus on SDSA, the neural network is presented, in this paper, with a simple mathematical model based on set theory, avoiding the simulation of the spiking neurons that were presented in [1] to realize the analogic framework. The first aim of this paper is to explain the software simulation of the network covering all the details of the algorithm. The second aim is to describe roughly how the digital pulsed model should work, without covering all the issues related to such an implementation. The mathematical model presented in this paper is the most efficient way to simulate the behavior of the SHARP neural model in the Von Neumann architecture.

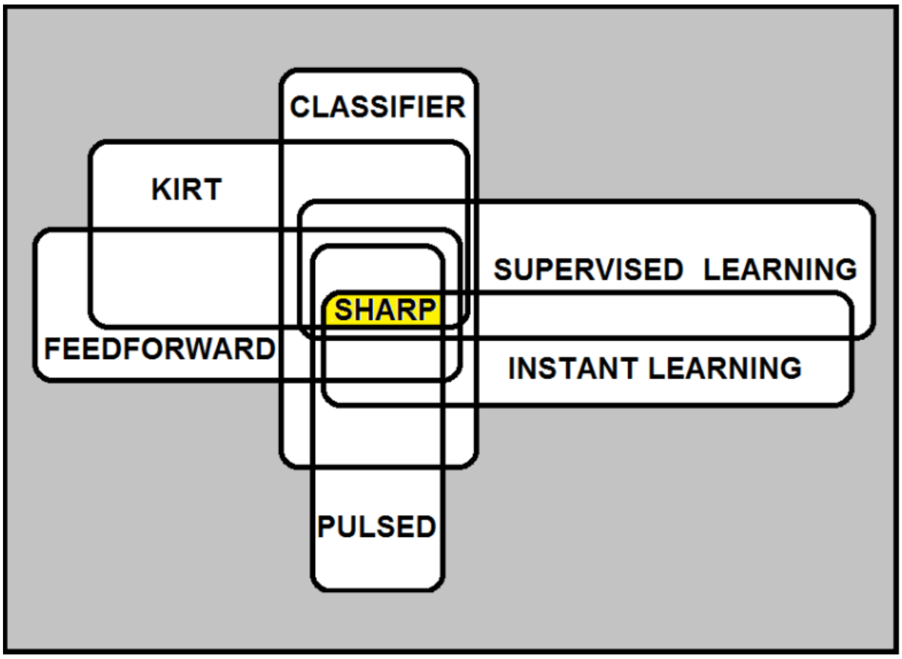


Fig. 1 The neural network model SHARP can be mapped in the intersection of the sets containing the classifiers and the feedforward neural networks learning in a supervised context. The network learns in one step and the recognition time, in the software simulation on a serial computer, is independent of the number of the learned patterns (KIRT = Knowledge Independent Recognition Time). The network can be implemented in a pulsed version in which the timing or the presence of the single spike is meaningful.

II. A Simplified Overview

This paragraph explains, with few words and many pictures, the very basic architecture and a simplified behavior of the network. The reader can deepen the knowledge of this paradigm in the following paragraphs that will explain, with mathematical rigor of the sets theory, the software simulation. The software simulation translates time delays to byte values, therefore, the following paragraphs will explain also some similarities and differences between the software simulation and the digital-pulsed implementation. Furthermore, the second part of the article will suggest some very speculative analogies of the model with the organization of the cerebral cortex. Fig.2 shows the initial status of the entire neural network with common inputs to the layers associated with different categories. Fig.3 shows the initial status of the connections within a single layer. Fig.4 is a photograph of the neurons resonating with the input stimulus. Fig.5 shows the first step of a supervised learning operation in a logistic regression context: the category layer is selected from the example. In Fig.6, within the selected layer, the synapses between the resonating neurons are updated. The generalization required by the training example (can be different for any feature) is obtained by updating the synapses of the neighboring neurons as shown in Fig.7. In order to simplify the visualization, the example of Fig.7 has a generalization value = 1. Fig.8 explains how to update synapses with generalization values > 1. Fig.9 shows the recognition of an input pattern (the network is not related to the previous learning example). Two layers (two categories) compete: the layer that sends all the pulses to the category neuron, with the shortest delay, wins. In the same picture, the firing condition of the neurons is explained.

The time elapsed between the generation of two pulses in the category layer can be considered the oscillation period of the network (in the software simulation this period is synthetized in the computation of all the layers) (Fig.10).

An optional feature of the network is the capability to learn correlations between input patterns presented in consecutive periods (in a software simulation these are consecutive examples in a training set) (Fig.11). The full explanation of this feature in a digital pulsed implementation is out of the scope of this document and some related issues could, probably, remain obscure. For example, one assumption for the correct behavior of the digital pulsed version is that the neurons of every layer are reset when a category neuron generates a pulse (Fig. 12). Therefore, at the be-

Copyright © Luca Marchese. ORC-ID: http://orcid.org/0000-0001-7903-7540 NCAGE: AK845

Email: luca.marchese@synaptics.org web: www.synaptics.org

ginning of an oscillation period the network starts with a common status of the neurons, although some pulses can be transiting on the delayed synapses.

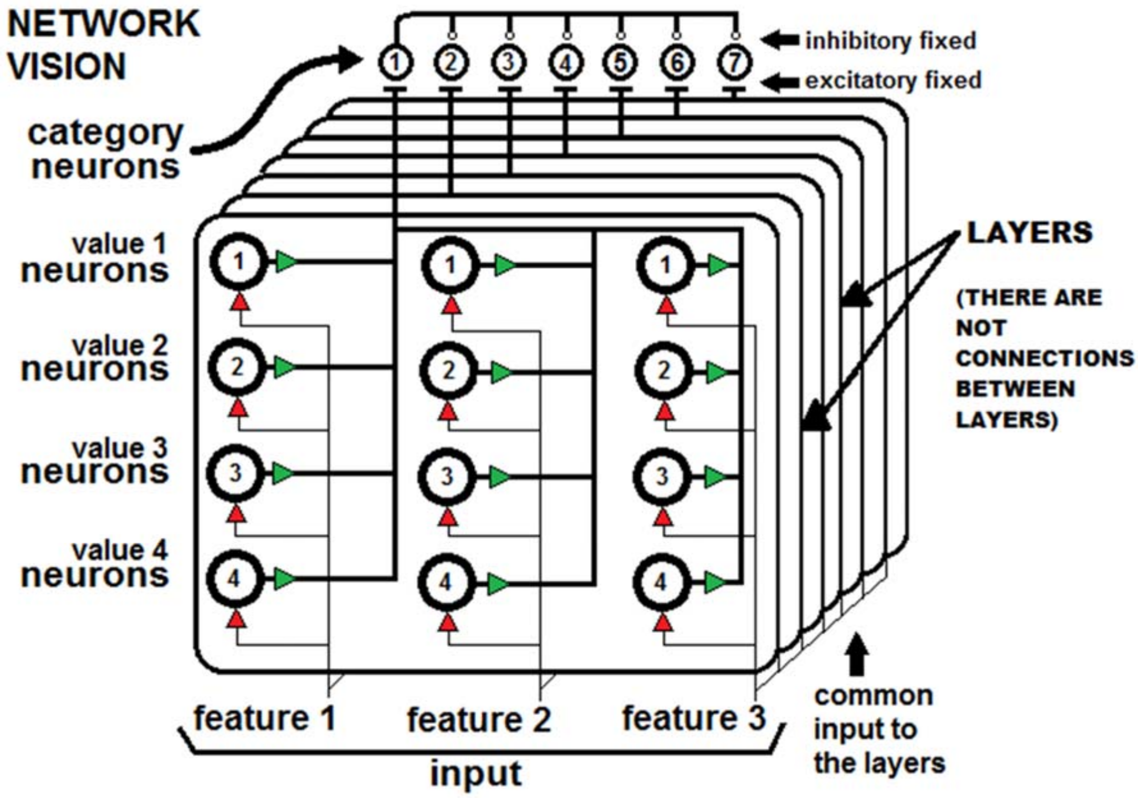


Fig. 2 The initial status of the entire neural network: the common input to layers must be noted. For simplicity the picture does not show the connections between neurons in the same layer.

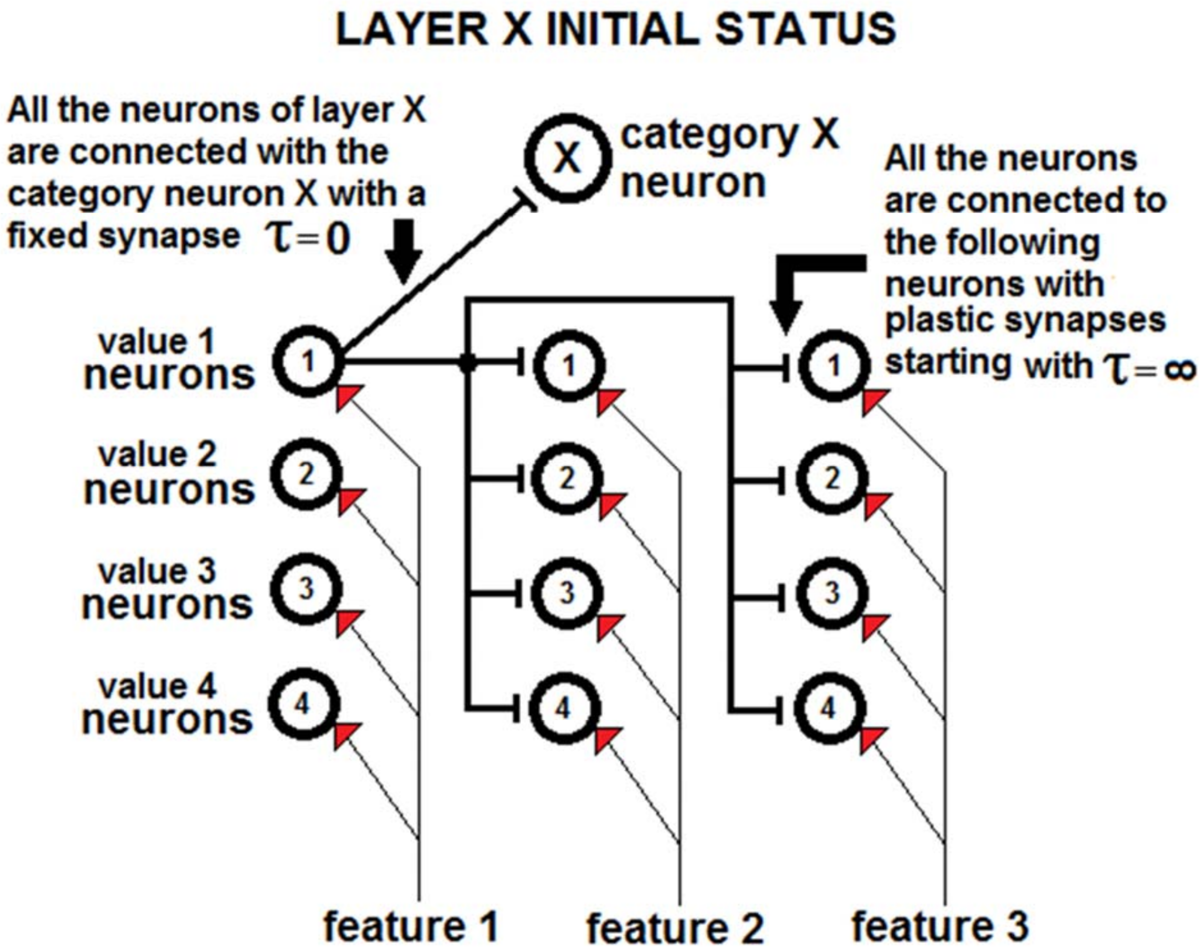


Fig. 3 The initial status of a single layer associated with a category.

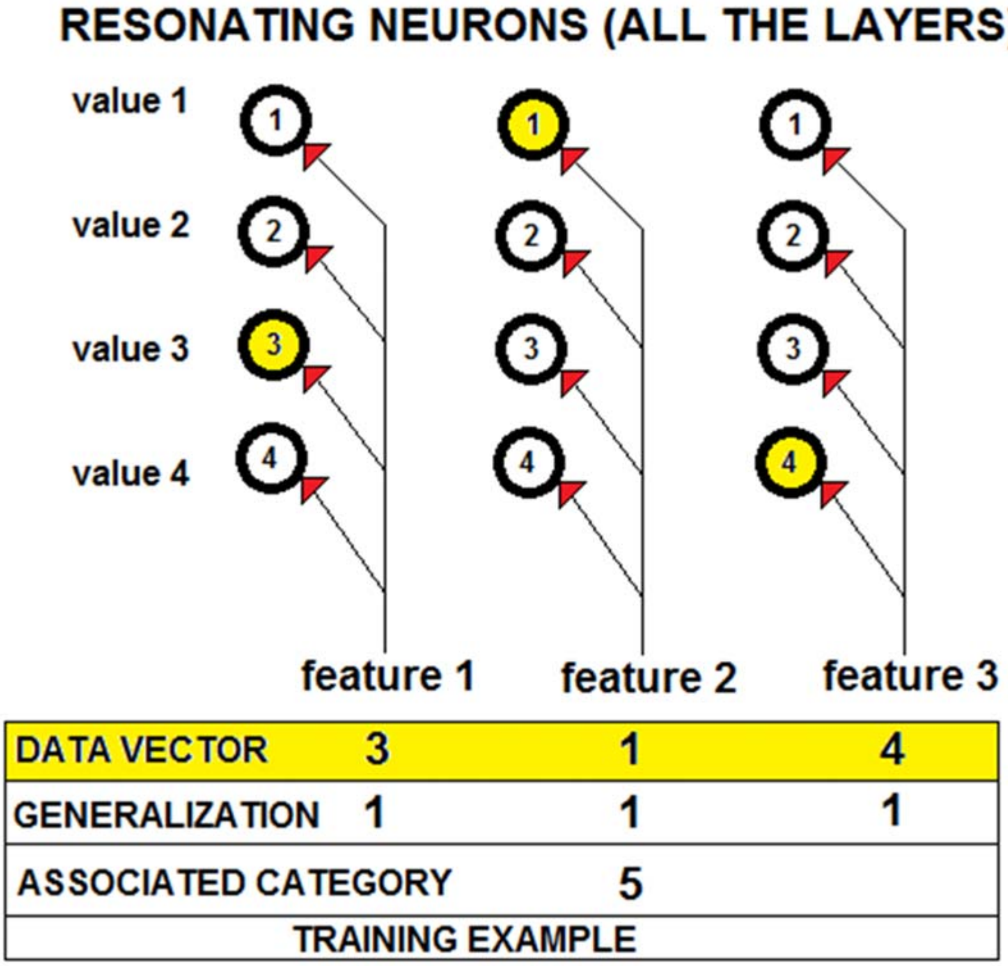


Fig. 4 The neurons, resonating with the input, are excited.

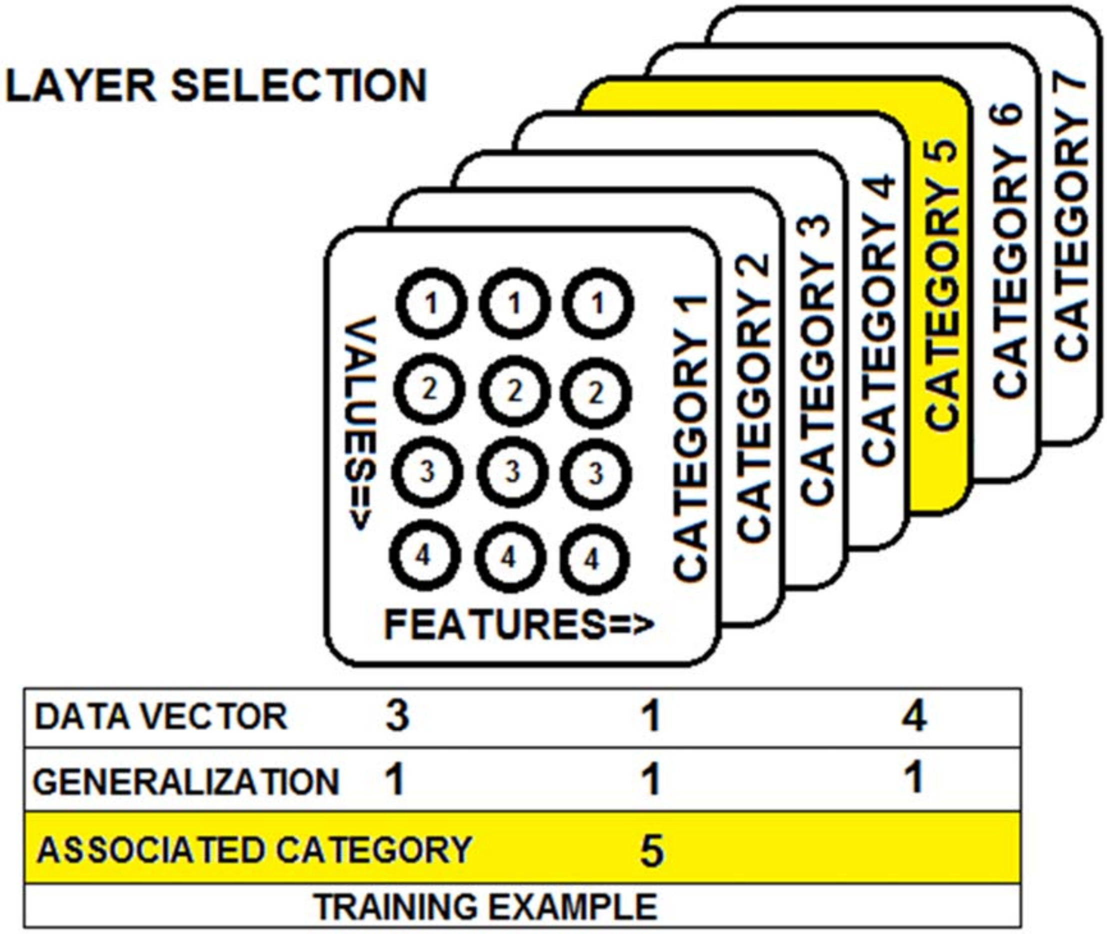
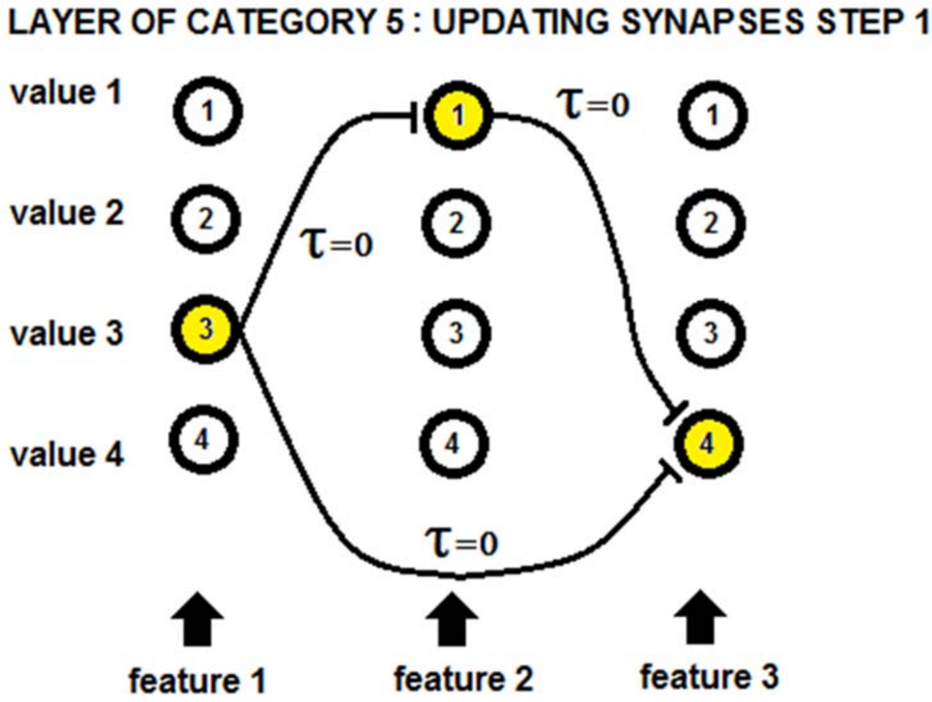


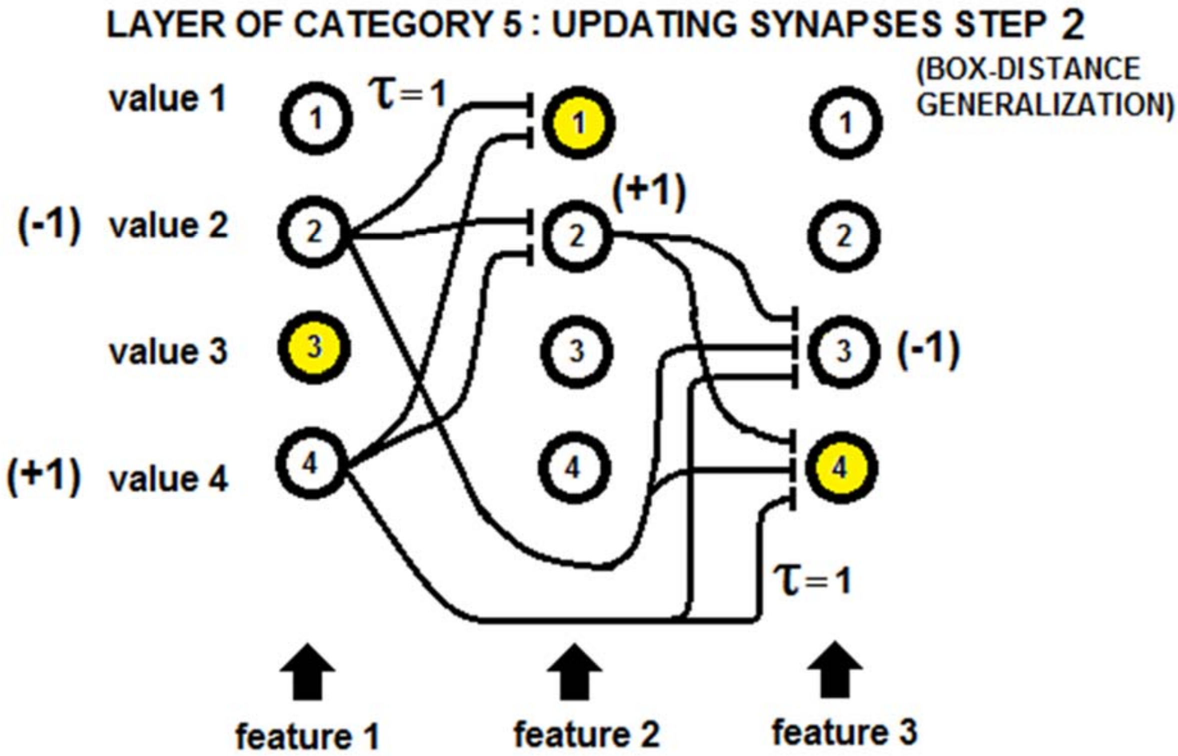
Fig. 5 The first supervised learning operation: the layer selection.



DATA VECTOR	3	1	4
GENERALIZATION	1	1	1
ASSOCIATED CATEGORY		5	
TRAINING EXAMPLE			

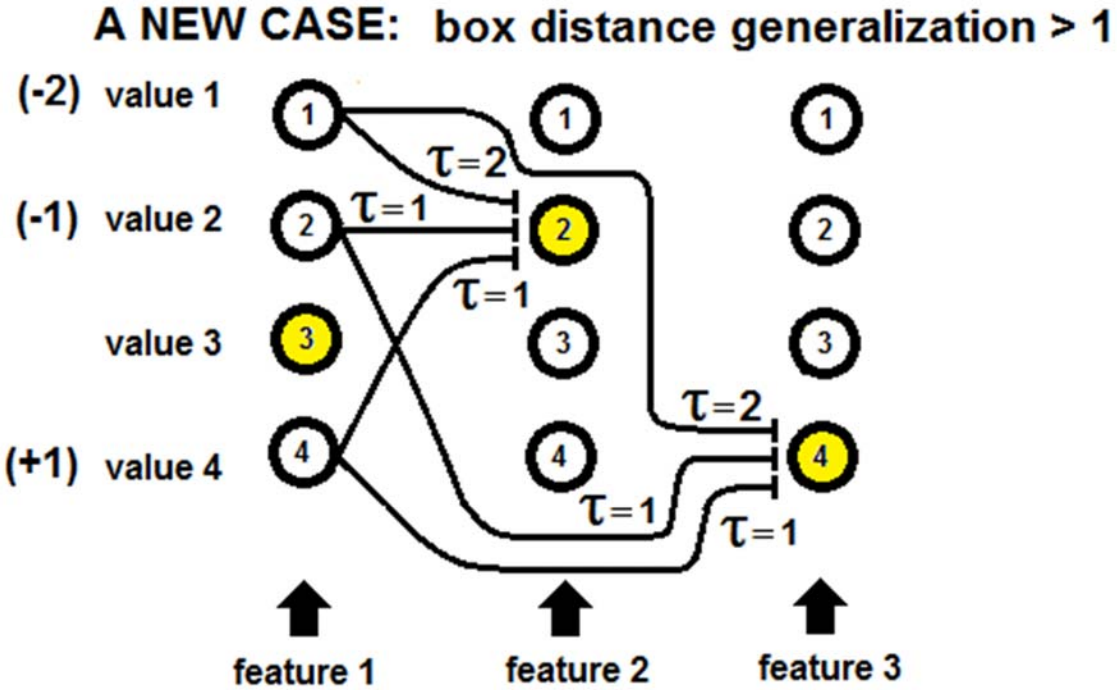
Fig. 6 The second supervised learning step: the Hebb rule is applied between resonating neurons.

Copyright © Luca Marchese. ORC-ID: http://orcid.org/0000-0001-7903-7540 NCAGE: AK845
Email: luca.marchese@synaptics.org web: www.synaptics.org



DATA VECTOR	3	1	4
GENERALIZATION	1 (±1)	1 (±1)	1 (±1)
ASSOCIATED CATEGORY		5	
TRAINING EXAMPLE			

Fig. 7 The third supervised learning step: the Hebb rule is applied between neighboring neurons.



DATA VECTOR	3	2	4
GENERALIZATION	2(±2)	0	0
ASSOCIATED CATEGORY		5	
TRAINING EXAMPLE 2			

in oredr to simplify the picture the generalization of the features 2 and 3 has been set to 0 in the example

Fig. 8 The third supervised learning step: the Hebb rule is applied between neighboring neurons.

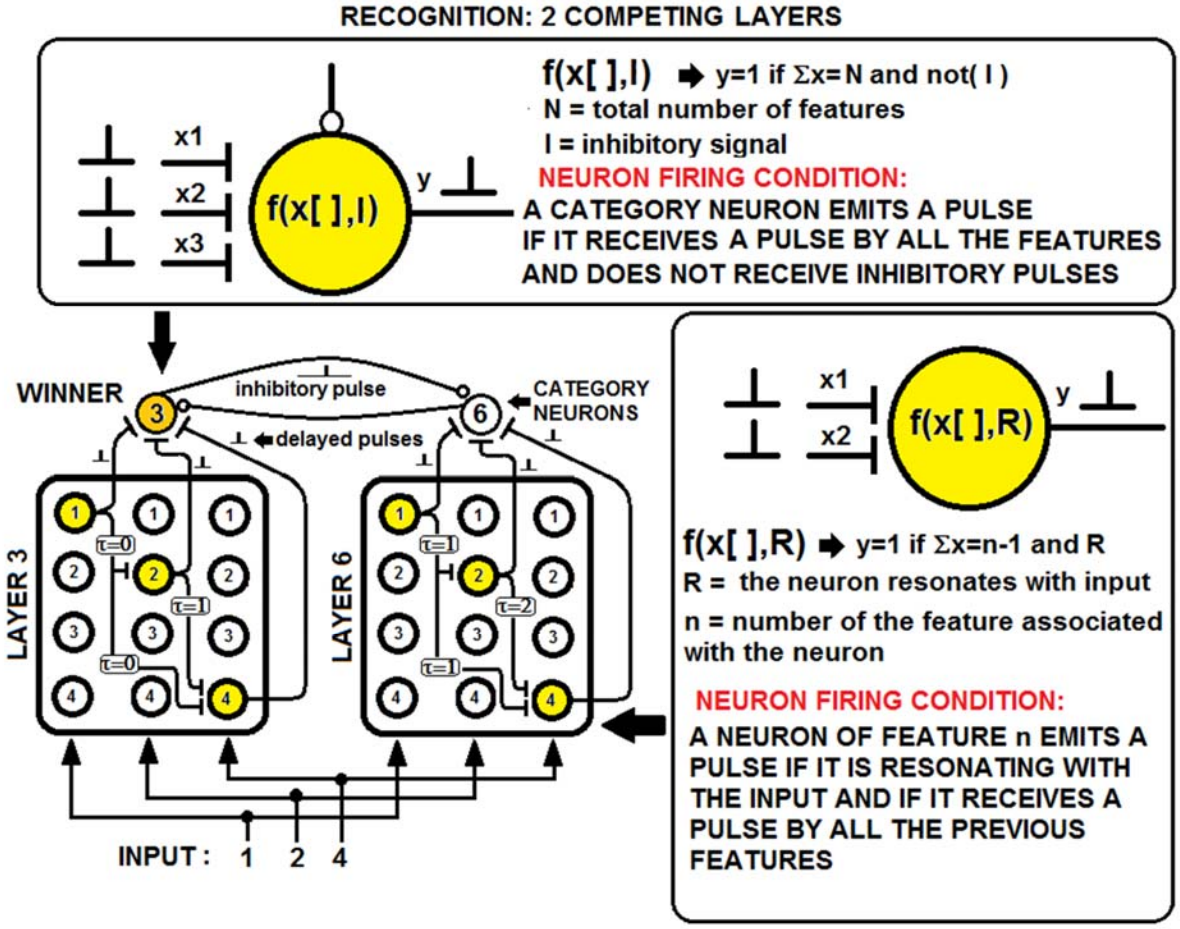


Fig. 9 The recognition phase: the delays of synapses produce different pulses sequences to the category neurons. The first category neuron that receives all the pulses fires and sends an inhibitory pulse to all the other category neurons. By disabling the WTA in the category layer, the category neurons emit spikes with a delay inversely proportional to the recognition strength (in the sw simulation the result is an array of bytes).

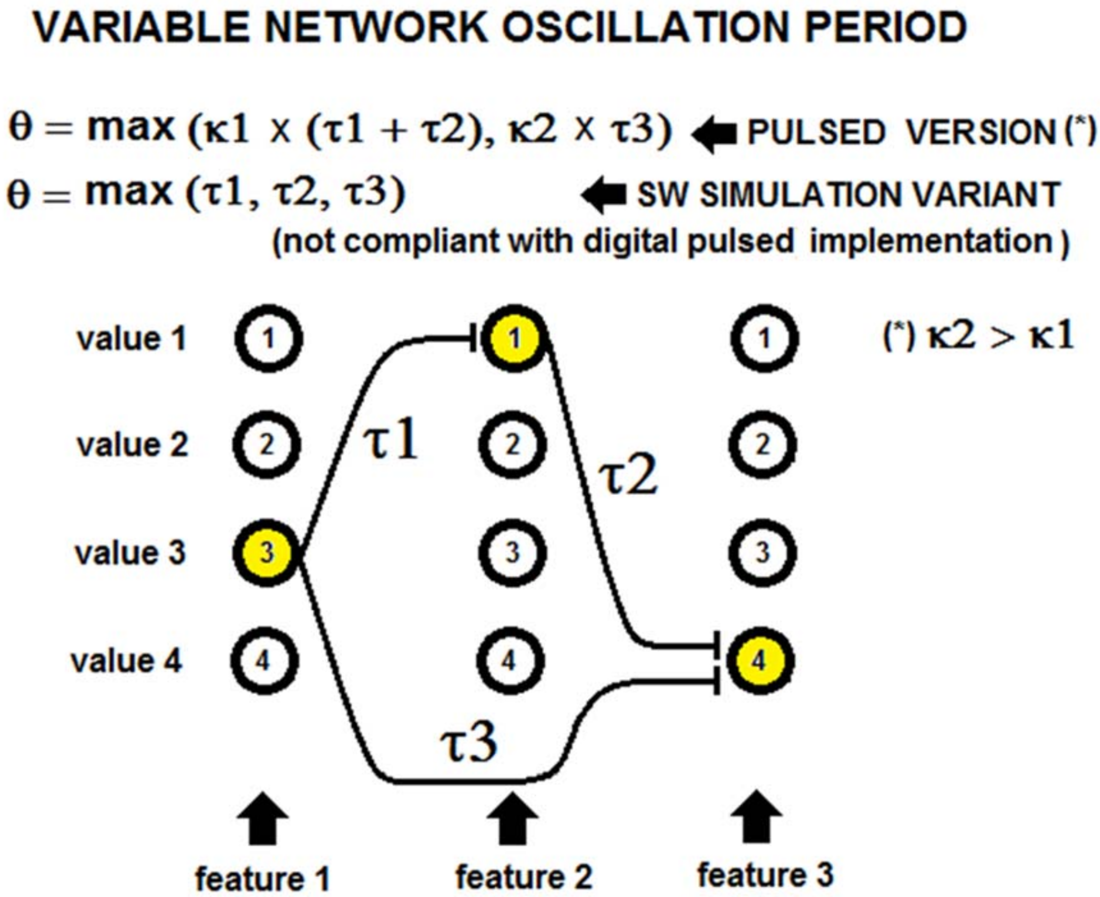


Fig. 10 The network has an oscillation period that coincides with the delay between two consecutive pulses generated by the category neurons. This period has length proportional to the sum of the delays of the synapses addressed by the current input stimulus.

Email: luca.marchese@synaptics.org web: www.synaptics.org

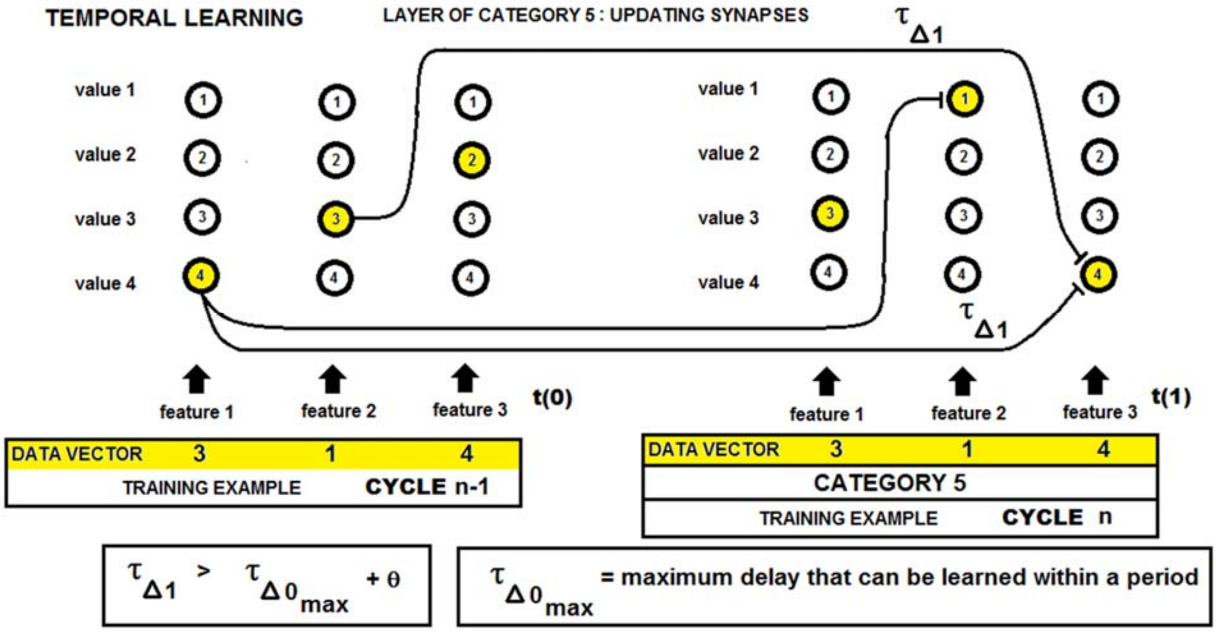


Fig. 11 The optional behavior is the Spatiotemporal Entangled Memory: the learning works by connecting neurons that resonate with input stimuli over consecutive periods. The learned delay is the sum of an offset that is multiple of the maximum period and a delay that is always larger than the maximum delay learned in the same period.

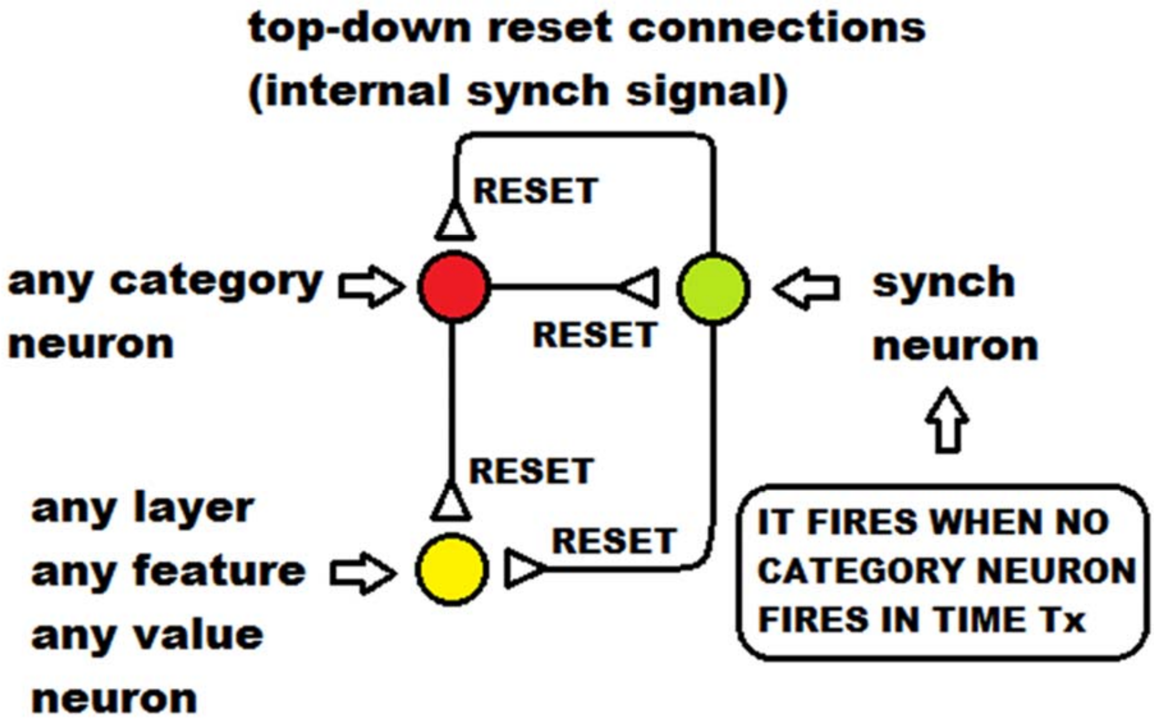


Fig. 12 The status of all the neurons needs to be reset when a category neuron fires or when a maximum time is elapsed from the last reset. Each category neuron sends a reset signal to all the neurons in the network. If no category neuron is activated, a synch neuron reaches spontaneously a firing threshold after a specified time and sends a reset signal to all the neurons in the network.

III. Architectural Description

The architecture of the network is a three-dimensional matrix for which the three dimensions represent, respectively, a mini-column, a macro-column and a cortical area network. Each mini-column is linked with a specific feature of the input stimulus and contains neurons resonating with specific values of the feature. The RF neurons in the mini-column are competitive modules (CM) working with a controlled WTA (Winner Takes All) mechanism. A macro-column is composed of mini-columns linked through synaptic connections that build paths representing configurations of the learned input patterns. Fig. 13 shows a macro-column and the path of synaptic connections required to complete the recognition of a specific pattern. One macro-column exists for each category. Specific configurations of the stimulus are associated with different categories during the learning process. The array of macro-columns constitutes the third dimension and completes the cortical area network (Fig. 13 and Fig. 14). The macro-column has feedforward connections: the axons originate from each mini-column and reach all subsequent mini-columns. The incremental flow of the feedforward connections suggests the term “systolic”, with reference to the application of the Hebb rule. The resonance of the neurons in the mini-columns is defined as “agnostic” because it is determined by the single value of the single feature and not by a vector of features/values as in Adaptive Resonance Theory [6]. During the supervised learning activity, the WTA within the mini-column is modulated by an inhibitory factor, which is inputted together with the input value. Therefore, the Hebb rule is applied to the neighboring neurons of the most resonating neuron (Fig. 15) and shapes the generalization property of the neural network in a Hyper-Cube Influence Field (LSUP/Box-Distance) [7]. The sides of the Hyper-Cube can have different magnitudes (Fig. 16 and Fig. 17). The bottom side of Fig.16 shows a very speculative example of a circuit realizing a mini-column with RF and IF neurons. The role of the SL (Supervised Learning) neuron is to trigger a learning task and partially inhibit the activity of the WTA neuron. In a pulsed version, the inhibition of the WTA would be the key for the Box-Distance generalization.

The firing condition of a mini-column neuron (explicitly included the resonance of the RF neuron) is computed in (1).

$$\left((f_m \approx f) \wedge \left(\int_{t=0}^{T} \sum_{x=1}^{n-1} s_{xc} \geq n-1-LNUM\right)\right) \Rightarrow s_{nmc} = 1; 0 - otherwise$$

$$\tau_S = \varepsilon \times \left(n-1-\int_{t=0}^{T} \sum_{x=1}^{n-1} s_{xc}\right) \tag{1}$$

where: f = input doublet frequency; s = spike presence;

n = feature index; m = value index; c = category index

$LNUM$ = maximum number of missing spikes from previous mini-columns

T = macro-column total processing time; τ_S = delay of the spike; ε = constant

In the analogic model [1], the mini-column neurons resonate with a doublet of spikes. The doublet frequency is used in equation (1) to calculate the firing condition. In the software simulation, the frequency is represented by the value of a byte. From this point, the author will speak about “software emulation” instead of “software simulation”. Indeed, the complex analogic behaviors described in [1] are emulated with a simple algorithm. The firing condition of a category neuron (including WTA) is computed in (2). The formulas (1) and (2) are not used in the software emulation, but are the base for a digital pulsed realization.

$$\left(\int_{t=0}^{T} \left(\sum_{n=1}^{N} s_{nc} - \left(LNUM + 1 \right) \times \sum_{c'=1;c'\neq c}^{C} s_{c'} \right) \geq N - LNUM \right) \Rightarrow s_c = 1; 0 - otherwise$$

$$\tau_S = \varepsilon \times \left(N - \int_{t=0}^{T} \sum_{n=1}^{N} s_{nc} \right) \qquad (2)$$

where: N = total number of features; C = total number of categories or macro-columns

The matrix of the forward excitatory connections is composed of the matrices of the feature values.

$$\phi = \begin{bmatrix} 0 & \varphi_{1,2} & \cdots & \varphi_{1,(m-1)} & \varphi_{1,m} \\ 0 & 0 & \cdots & \varphi_{2,(m-1)} & \varphi_{2,m} \\ \cdots & \cdots & 0 & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & \varphi_{(m-1),m} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \qquad (3)$$

where: m = number of features

Each element is a matrix of the feature values:

$$\varphi = \begin{bmatrix} \omega_{1,1} & \omega_{1,2} & \cdots & \omega_{1,n} \\ \omega_{2,1} & \omega_{2,2} & \cdots & \omega_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \omega_{n,1} & \omega_{n,2} & \cdots & \omega_{n,n} \end{bmatrix} \qquad (4)$$

where: n = number of values describing a feature

Each element of φ is a vector of values ρ (inversely proportional to the delay of the synapse) associated with categories:

$$\omega = \begin{bmatrix} \rho_1 & \rho_2 & \cdots & \rho_j \end{bmatrix} \qquad (5)$$

where: j = number of categories

The resulting matrix of the feedforward excitatory connections is:

Email: luca.marchese@synaptics.org web: www.synaptics.org

$$\phi = \begin{bmatrix} 0 & \begin{bmatrix} \omega_{1,1} & [\rho_1 \quad \rho_2 \quad \cdots \quad \rho_j]_{1,2} & \cdots & \omega_{1,n} \\ \omega_{2,1} & \omega_{2,2} & \cdots & \omega_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \omega_{n,1} & \omega_{n,2} & \cdots & \omega_{n,n} \end{bmatrix}_{1,2} & \cdots & \varphi_{1,(m-1)} & \varphi_{1,m} \\ 0 & 0 & \cdots & \varphi_{2,(m-1)} & \varphi_{2,m} \\ \cdots & \cdots & 0 & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & \varphi_{(m-1),m} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{6}$$

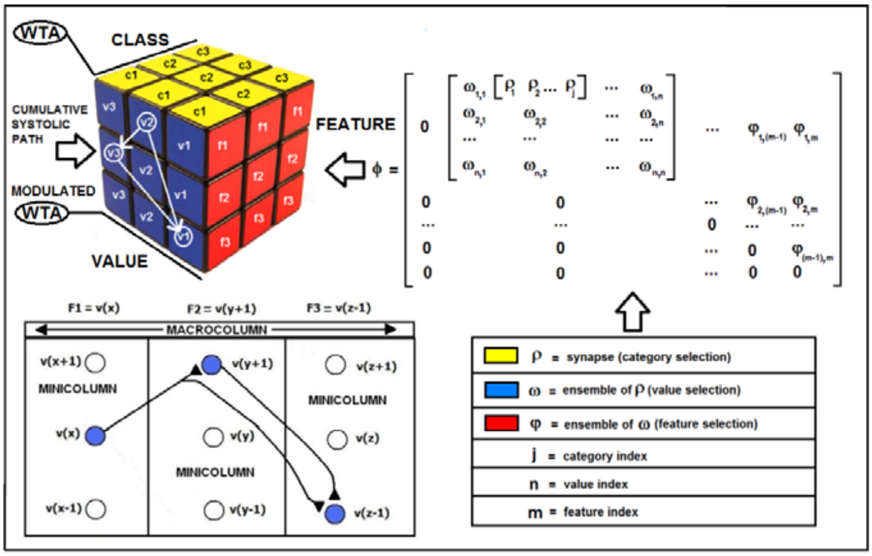


Fig. 13. Three cortical mini-columns in one cortical macro-column are shown in the bottom-left side of the picture. The input pattern F1=v(x), F2=v(y+1), F3=v(z-1) has been learned and the synapses between the pairs of neurons v(x) to v(y+1), v(x) to v(z-1), and v(y+1) to v(z-1) have been reinforced through the Hebb rule. When the same pattern is presented to the network, the neuron v(x) resonates with the input value and emits a spike. The neuron v(y+1) resonates with the input value and emits a spike, receiving a consensus spike from mini-column 1. The neuron v(z-1) resonates with the input value and emits a spike, receiving two consensus spikes from mini-column 1 and mini-column 2, respectively. Every macro-column is associated with a category neuron. The path ***v2(f1),v3(f2),v1(f3)*** is shown in the macro-column in ***c1*** on the blue side of a cube that represents the tridimensional matrix Feature-Value-Class. The matrix of all of the features (fx) and values (vx) represents a macro-column. The full tridimensional matrix ***fx-vx-cx*** represents a cortical area network. In this picture, the fourth dimension (time) involved in the behavior of the complete neural model is not represented.

Email: luca.marchese@synaptics.org web: www.synaptics.org

Neural Network Model	Recognition time independent of the number of learned patterns	Learning in one step	Readable relations / Rule Extraction (NN is not a Black Box)
MLP (Backprop)	YES	NO (many cycles)	NO
RBF (L1(City-Block) / L-SUP)	NO	NO (few cycles)	YES
SHARP	YES	YES (one cycle)	YES

Table 1. A comparative table between SHARP, MLP and RBF that analyzes three important characteristics of the neural networks. The first characteristic is the independence of the recognition time from the number of the learned patterns; RBF, independently of the learning algorithm, requires that the input vector be compared with all of the stored prototypes: the matching pattern could be the first or the last, but the average time is half the size of the database and therefore is proportional to the number of stored prototypes. The second characteristic is the capability to learn in one step. It is well known that MLP with backpropagation requires many cycles (epochs) on the entire dataset. RBF could require a few cycles to re-learn patterns that have been lost due to a Neuron Influence Field reduction triggered by a class-mismatch (the equivalent operation in supervised ART would be increasing Vigilance). The third characteristic is the possibility of extracting rules from the trained neural network. MLP is a Black Box. SHARP has readable relations, although the operation of rule extraction requires a full scan of the synapses. During the learning process it is possible to build a database of rules that reflects the behavior of the neural network.

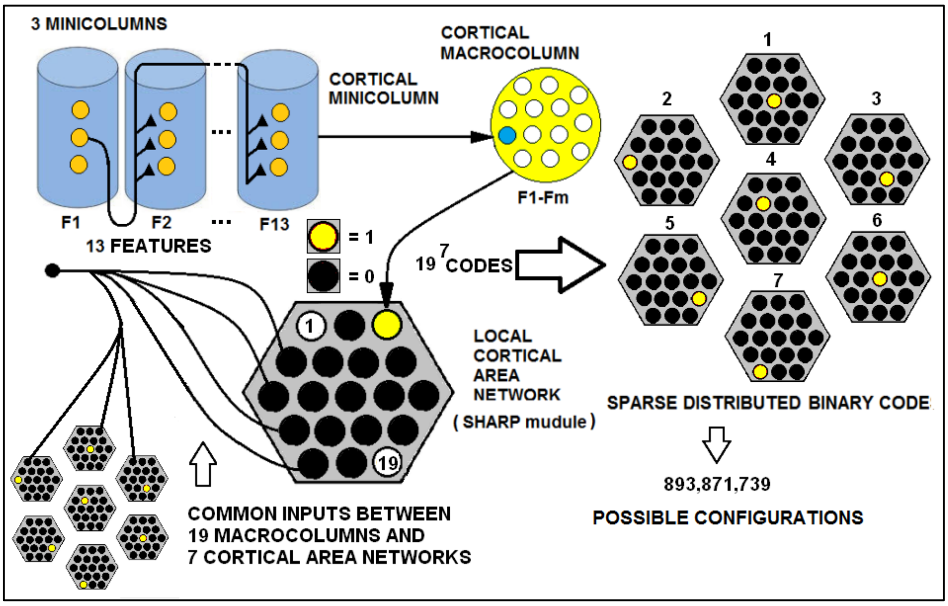


Fig. 14. In this representation of the SHARP neural network, it is possible to distinguish the components of the mini-columns from the macro-columns and the cortical area network. In this picture, each macro-column of seven cortical area networks (seven SHARP modules) receives the same inputs. Every macro-column is connected with a category neuron that represents an item of a sparse distributed binary code (SDC). In this picture, a SHARP module manages 19 categories.

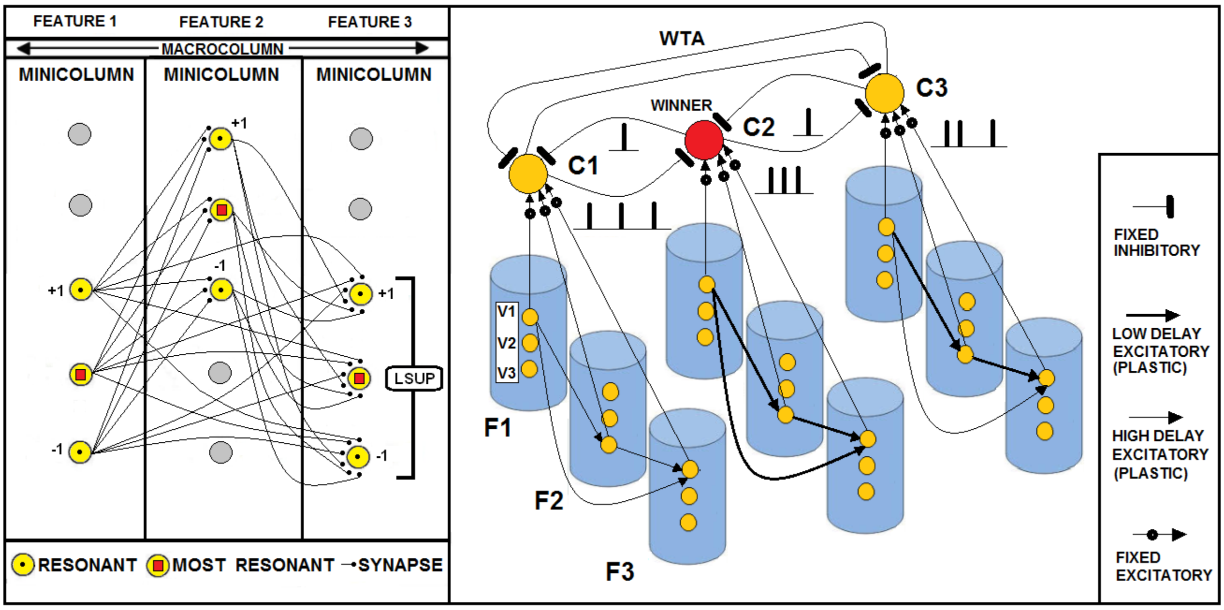


Fig. 15. LEFT: This picture illustrates a macro-column with three mini-columns in which the Hebb rule is applied to the neighboring neurons of the most resonant neuron. The applied LSUP value is 1 (i.e., +/-1) in all of the mini-columns. The number of involved synapses for any neuron of mini-column 2 is equal to 2 * LSUP [mini-col 1] + 1 (=3), while the number of involved synapses for any neuron of the mini-column 3 is equal to 2 * LSUP [mini-col 1] + 1 * LSUP [mini-col 2] + 1 (=6). RIGHT: Three macro-columns composed of three mini-columns. The WTA on the category layer is driven by the delays in the mini-columns.

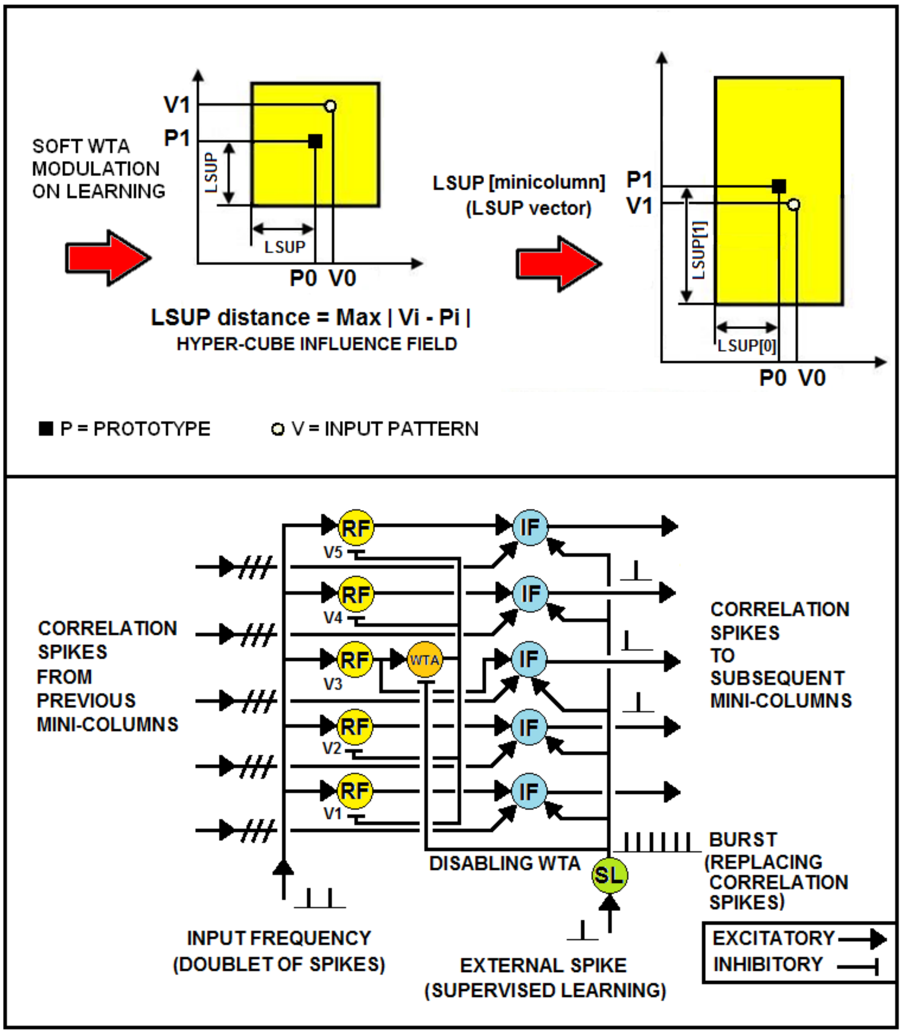


Fig. 16. TOP: The effect of the soft WTA modulation during the learning activity. Not only the most resonant neuron but also the neighboring (LSUP) neurons can fire. The picture also shows the difference between a fixed LSUP and the variable LSUP, which can be different for any mini-column (i.e., for any feature) and can be assigned dynamically during the learning process. BOTTOM: A speculative example of a possible implementation of a mini-column with Resonate and Fire (RF) neurons and Integrate and Fire (IF) neurons. A WTA neuron is present for each RF neuron and inhibits all of the other RF neurons. The SL neuron receives an external supervised learning spike and generates a burst of spikes which softly disable the WTA (generating LSUP generalization) and replace the correlation spikes from the previous mini-columns (in a pulsed implementation this behavior would be required in order to enable the resonating neurons to fire and then the Hebb rule, between firing neurons, could be applied).

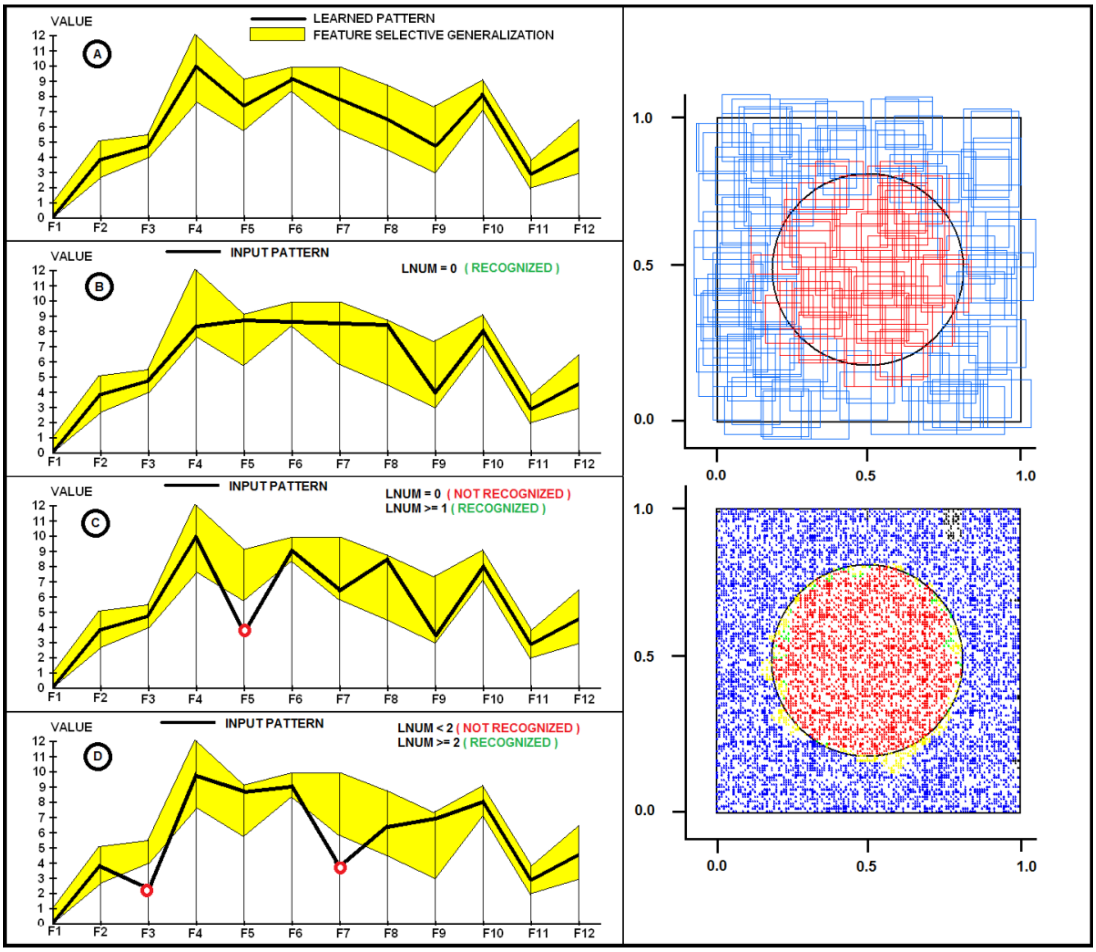


Fig. 17. (A) The picture shows the feature-selective generalization that can be learned with the input pattern. An LSUP generalization that can be learned for any single component of the input vector is extremely flexible. (B) This is an example of a pattern recognition based on only the LSUP generalization. (C, D) These are two examples of input patterns that are recognized or not recognized, depending on the LSUP and the LNUM generalization factor, which is set outside the learning activity as a parameter of the network. On the right side the "circle in the square" problem: the superimposed LSUP regions are resolved by the WTA in the category layer.

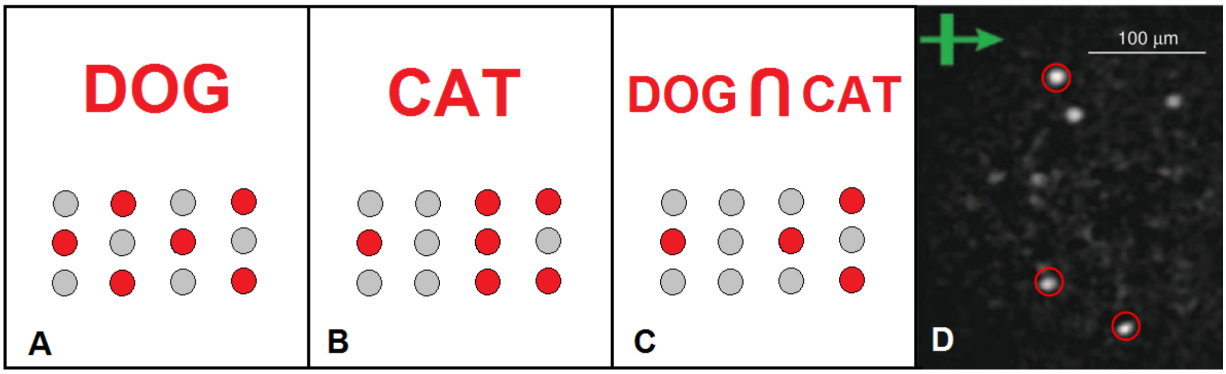


Fig. 18. An example of a sparse distributed code. Some elements used for the representation of the concept “dog” (A) are used for the representation of the concept “cat” (B), as shown in the intersection (C). (D) From Ohki et al. (2005) [16].

IV. Instantaneous Learning and Real-Time Recognition with SDSA

The SHARP neural network algorithm learns in a single step and recognizes a pattern in a timeframe that is independent of the number of learned patterns. Therefore, the algorithm can be executed on a Von-Neumann computer with constant performance as an RBF (radial basis function) [8][9] network executed on a correctly sized SIMD (single instruction multiple data) computer [10]. The formulas presented in this paper are simplified for the software emulation and are not completely compliant with the pulsed realization (the total delay of the macro-column is approximated to the maximum single delay). The recognition task is performed as in (7), (8) and (9):

$$L = \varnothing; \forall j P_j = \varnothing; \forall j \forall x \left(M_{jx} = \varnothing \right)$$

$$\forall j \forall x \left(\forall y \neq x \left(\rho_j \in \omega_{inp(x),inp(y)} \in \varphi_{x,y} \right) = 0 \Rightarrow M_{jx} = M_{jx} \cup \{ y \} \right)$$

$$\text{given} \quad f : x \to \# M_{jx} \quad \forall j \forall i \leq LNUM \left(\left(\# P_j < LNUM \right) \Rightarrow P_j = P_j \cup \left\{ \arg\max_{x=1, x \notin P_j}^{N} \left(f(x) \right) \right\} \right)$$

$$\forall j \left(\left(\# P_j > LNUM \right) \Rightarrow \left(\chi_j = 0; L = L \cup \{ j \} \right) \right)$$

where: (7)

$x, y, i \in X = \{1, 2, ..., N-1, N\}; M_{jx} \subseteq X; P_j \subseteq X; \# M =$ cardinality of M;

$L \subseteq J = \{1, 2, ..., C-1, C\}; N =$ number of mini-columns;

$C =$ number of categories; $\chi_j =$ recognition strength of category j;

$inp(x) =$ input value of mini-column x

The recognition strength (inversely proportional to the maximum delay in the systolic path) for the remaining categories is computed:

$$\forall \left(j \notin L \right) \chi_j = \frac{\min_{x=1}^{N-1} \left(\min_{y=x+1}^{N} \left(\rho_j \in \omega_{inp(x),inp(y)} \in \varphi_{x \notin P_j, y} \right) \right)}{\# P_j + 1} \tag{8}$$

The denominator in (8) emulates the τ_s in (1).

The WTA is applied:

given $f : j \to \chi_j$

$W = \arg\max \left(f(j) \right);$

$\left(\# W = 1 \right) \Rightarrow class = j \in W;$ (9)

$\left(\# W = 0 \wedge \chi_j = 0 \right) \Rightarrow$ not recognized;

$\left(\# W > 1 \right) \vee \left(\# W = 0 \wedge \chi_j > 0 \right) \Rightarrow$ recognition uncertanty;

The learning task is performed as in (10):

$$\forall\left(1\le x<N\right)\forall\left(y>x\right)\forall\left(\left(V_{ref(x)},V_{ref(y)}\right)\ni\left(\left|V_{ref(x)}-V_{inp(x)}\right|\le LSUP_x\right)\wedge\left(\left|V_{ref(y)}-V_{inp(y)}\right|\le LSUP_y\right)\right)$$

$$\rho_j\in\omega_{V_{ref(x)},V_{ref(y)}}\in\varphi_{x,y}=$$

$$\max\left(\left(\rho_j\in\omega_{V_{ref(x)},V_{ref(y)}}\in\varphi_{x,y}\right),\left(\rho_{MAX}-\left(\frac{\rho_{MAX}}{LSUP_{MAX}}\right)\times\max\left(\left|V_{ref(x)}-V_{inp(x)}\right|,\left|V_{ref(y)}-V_{inp(y)}\right|\right)\right)\right)$$

where:

$\wedge$ = logic AND, $\ni$ = such that

x = starting feature, y = destination feature, N = number of features

V_{ref} = reference value of the mini-column neuron

V_{inp} = value of the feature in the training example (10)

$LSUP$ = LSUP associated with the feature in the training example

In (10) the synapse assumes a delay which is inversely proportional to the minimum level of resonance of the RF neurons associated with the pre-synaptic and post-synaptic IF neurons.
LSUP is supplied for any feature in the training example in order to emulate the presence of an external signal [1] that triggers the learning activity and inhibits the WTA within the mini-columns. The inhibition of the WTA enables neighboring neurons to activate and the application of the Hebb rule generates the LSUP generalization (Fig. 16). An alternative behavior of the learning process automatically manages the LSUP on the basis of the familiarity (recognition strength) of the input stimulus. Formulas (7) with LNUM=0 (or simply with $P=\varnothing$), (8) and (9) are applied. LSUP is then computed with:

$$\left(\left(\#W=0\wedge\chi_j=0\right)\vee j_{win}\ne j_D\right)\Rightarrow\forall\left(x\le N\right)LSUP_x=LSUP_{MAX}$$

$$\left(j_{win}=j_D\right)\Rightarrow\forall\left(x\le N\right)LSUP_x=LSUP_{MAX}\times\left(1-\left(1+e^{-\beta}\right)^{-1}\right) \quad (11)$$

where: $\beta=\chi_{win}-\left(\chi_{MAX}/2\right)$

j_D = desired category

In this case, the same LSUP is applied to all of the mini-columns. LSUP assumes the maximum value when the pattern is not recognized and the minimum value when the pattern is recognized with the maximum strength. As the network is working in a supervised learning context, if the winning category is different from the desired category, LSUP assumes the maximum value.

A second method to compute internally the LSUP with a specific value for any mini-column is to modify (11) as follows.

$$\beta_x = \max_{y=1}^{N}\left(\rho_j \in \omega_{inp(x),inp(y)} \in \varphi_{x,y\neq x}\right)$$
$$\left(j_{win} = j_D\right) \Rightarrow \forall\left(x \leq N\right) LSUP_x = LSUP_{MAX} \times \left(1 - \left(1 + e^{-\beta_x}\right)^{-1}\right) \tag{12}$$

Formula (12) computes, with probabilistic approximation, the distance of the feature input value from the value of the original prototype through the evaluation of the maximum synaptic strength (minimum delay) connecting the mini-column with all of the other mini-columns. The internal management of the LSUP implies the presence of intra-macro-column fixed inhibitory synaptic connections that modulate the WTA activity based on the familiarity of the input pattern.
Table 1 shows a comparison between the SHARP, the RBF and the MLP neural networks with respect to learning, recognition and rule extraction. Table 2 shows a comparison of the performances of SHARP and RBF on a serial computer using a large dataset.

There is increasing evidence that the sparse distributed code (SDC) plays a fundamental role in the cortex and, perhaps, in many other structures of the brain. One important advantage of the SDC over a localist code is that the number of unique patterns that can be stored is much larger than the number of representing neurons. In a distributed representation of an item of information, multiple elements collectively represent that item and each of the elements can be involved in the representations of other items. Distributed representations are often referred to as population codes. The SDC can be considered a special case of distributed representation in which a small part of the full ensemble of representing elements is involved in the representation of a particular item and any element can be involved in the representations of multiple items (Fig. 18).

		SHARP LSUP = 5 (μ =5) LSUP = 10 (μ =10)	RBF (LSUP) Min LSUP = 0 Max LSUP = μ	 Max LSUP = 90
RECALL	ID	10000	10000	10000
	UNC	0	0	0
	NID	0	0	0
	WID	0	0	0
VALIDATION μ =5 (noise up to 10% on the components)	ID	10000	10000	7267
	UNC	0	0	2732
	NID	0	0	0
	WID	0	0	0
VALIDATION μ =10 (noise up to 20% on the components)	ID	10000	10000	6812
	UNC	0	0	3187
	NID	0	0	0
	WID	0	0	0
VALIDATION μ =15 (noise up to 30% on the components)	ID	7810	10000	6764
	UNC	1147	0	3235
	NID	0	0	0
	WID	1043	0	0
TLT (1 CYCLE)		67 mS/216 mS	340 mS	320 mS
TRT		1.0 mS	400 mS	390 mS
APLT		11 µS/40 µS	34 µS	32 µS
APRT		**100 nS**	**40 µS**	**39 µS**
CPN		NA	10000	9341
CYCLES		1(NA)	1	6

Table 2. The table shows the results of the comparative test between SHARP and RBF with a training database of 10000 pseudo-randomly generated patterns with minimal constraints related to the associated class. The validation database is composed of patterns derived from the learning patterns, adding noise on any component limited by µ (c – µ < c' < c + µ). ID = correctly identified; UNC = identified with uncertainty; NID = not identified; WID = wrongly identified. TLT = Total Learning Time; TRT = Total Recognition Time; APLT = Average Pattern Learning Time; APRT = Average Pattern Recognition Time; CPN = Committed Prototype Number (applicable to RBF). All of the computed times are related to the effective time required by the learning/recognition algorithm: the overhead related to all of the other operations required (read / write / preprocessing) that are congruent for both the neural paradigms have been excluded. The test has been executed on an Intel® Core™ i5-3320M CPU @ 2.60 GHz with 4 GB RAM. In the table, the differences in the averaged pattern recognition time (APRT) between SHARP and RBF are highlighted in yellow: SHARP is 400 times faster than RBF and the difference in performance should grow as the number of the learned patterns grows. The recognition time has been improved compared with the same test performed in [1] by removing useless operations in the software emulation. In the SHARP column, the minimum learning time is associated with LSUP = 5 and the maximum learning time is associated with LSUP = 10. In this software emulation, each mini-column is composed of 128 neurons (80-120 is a biologically plausible range) and each macro-column is composed of 64 mini-columns (50-100 is a biologically plausible range).

Email: luca.marchese@synaptics.org web: www.synaptics.org

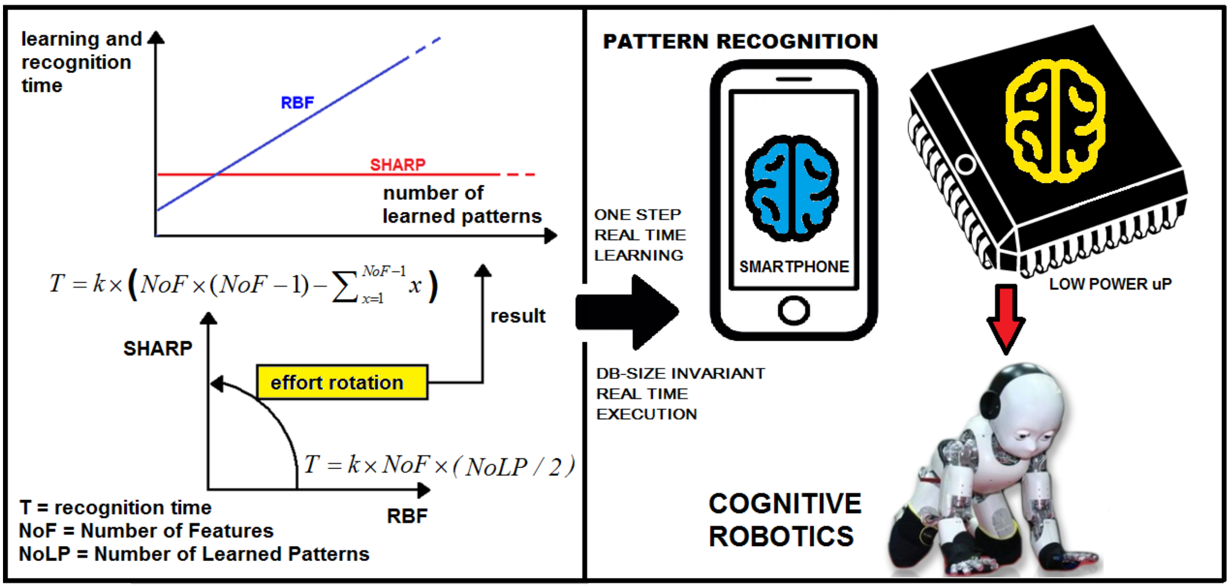


Fig. 19. The picture (top) compares the recognition time of a SHARP neural network with that of an RBF neural network on a serial computer: for the SHARP neural network, the recognition time is independent of the number of learned patterns. The picture (bottom) shows the SHARP transformation that changes the relationship between the pattern recognition time and the variables “number of features” and “number of learned patterns”. Thanks to the SDSA property, the NN can search for the best matching pattern in a time that is independent of the number of possible solutions (i.e., the learned patterns). Indeed, this is the behavioral result of the SHARP transformation that, through the incremental correlation between the mini-columns, enables the SDSA to realize a quasi-quadratic proportionality between the recognition time and the number of features. The picture of the iCub humanoid robot developed at IIT (Istituto Italiano Tecnologie) is from icub.org.

Other models linking mini- and macro-columns with the use of the SDC have been presented in other works. G.J. Rinkus [11][12][13] presented a model that explains with a scientifically relevant approach the biological plausibility of the SDC in the cortex with many references to experimental and theoretical studies [14][15]. In the model designed by Rinkus, the macro-column is proposed to store information in the form of SDC, and the mini-column (specifically, its L2/3 of pyramidals) is proposed as a WTA CM, the purpose of which is to enforce the sparseness of the macro-columnar code. The target of this study was a brain-inspired pattern recognition engine and in this paper the author underlines the computational efficiency of the SDC and explains how this type of information coding works in the SHARP neural network model. The author wants to distinguish between the elements that are involved in the storage of the learned patterns and the elements that are involved in the external representation of the category associated with the learned patterns: the terms SDS (sparse distributed storage) and SDC (sparse distributed code) are used to refer to these two different concepts. In the SHARP architecture, each

mini-column represents a feature, and the ensemble of mini-columns constitutes a macro-column. The ensemble of macro-columns associated with different category neurons is proposed as a cortical area network. Each mini-column is organized in a WTA mode, and the competing elements are the single neurons inside each mini-column. The mini-column contributes to the sparseness of the information by distributing the values of the single feature of the input stimulus on different neurons. The macro-column distributes the information along the dimension of the features of the input stimulus. An information item is represented by an ensemble of the synaptic connections between each mini-column and the subsequent mini-columns.
SHARP recognizes an input stimulus by addressing only the involved synapses. This process is the functional implementation of the SDSA (stimulus-driven synaptic addressing) property, which was also called the SIASP (similar inputs address similar paths) property in [17]. The "virtual parallelism" is generated by the SDSA that works on the accumulation of the correlations (carried by spikes) between the mini-columns and produces the computational effort transformation represented in Fig. 19. The recognition effort is no longer proportional to the number of learned patterns (as in RBF): the effort required to scan the database of prototypes is replaced by the analysis of the correlations between the mini-columns (13). The time required for learning a new pattern is still independent of the number of learned patterns but it is biased by a factor that is function of the LSUP values of any mini-column (14). The learning time rises with the LSUP value because a larger number of synapses must be updated.

$$T_{RBF(average)} = k_1 \times \left(NoF \times \left(NoLP \div 2 \right) \right)$$
$$T_{SHARP} = k_2 \times \left(NoF \times \left(NoF - 1 \right) - \sum_{x=1}^{NoF-1} x \right) \tag{13}$$

where:

$NoF =$ number of features $NoLP =$ number of learned patterns

$$T_{SHARP(LEARN)} = k_3 \times \left(\sum_{x=1}^{NoF-1} \sum_{y=x+1}^{NoF} \left(2 \times LSUP_x + 1 \right) \times \left(2 \times LSUP_y + 1 \right) \right) \tag{14}$$

SHARP can work with binary input patterns. In this case, in each mini-column there are multiple "lattice sections" with two neurons associated with the Boolean values 0 and 1. Each section is associated with a binary feature, therefore each mini-column receives inputs from multiple binary features. The learning rule (10) is still valid but the generalization parameter LSUP must be set to 0. The generalization is performed only through the LNUM parameter. The recognition rules (7), (8) and (9) are still correct although the synapses become weightless (0, Max) and the

Email: luca.marchese@synaptics.org web: www.synaptics.org

numerator of (8) can be directly replaced with ρ_{MAX} . An example of a “lattice section” of a binary macro-column composed of five mini-columns is shown in Fig. 20. The model based on binary mini-columns can be used in deep learning applications in order to process SDCs.

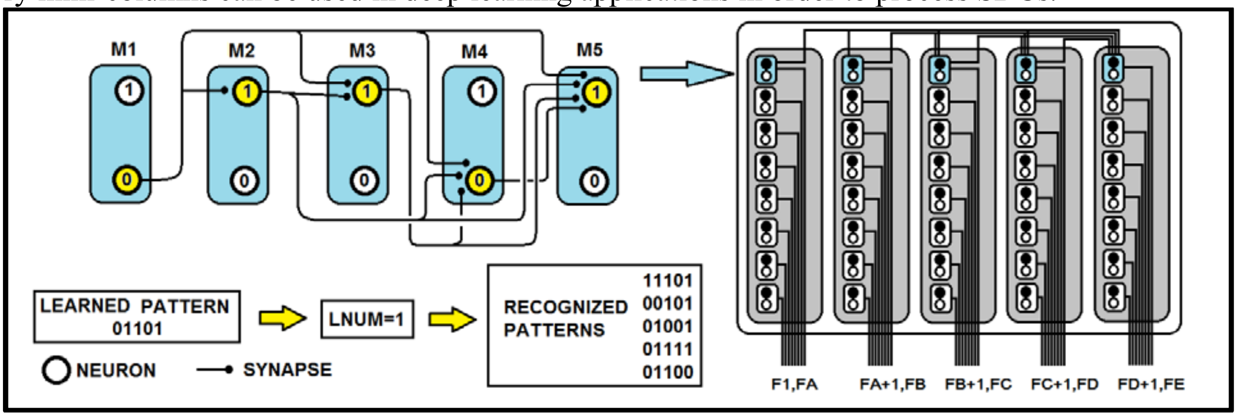


Fig. 20. The picture shows a lattice section of a binary macro-column composed of five mini-columns. The learning process of one pattern has modulated the represented synapses with the max value. The generalization, during the recognition process, is performed through the LNUM parameter (i.e., LNUM=1 means that each resonating neuron in the mini-column *n* can fire with *n-2* spikes from the previous mini-columns instead of *n-1*). In the proposed software emulation, each mini-column has 128 neurons and can be divided in 64 lattice sections, therefore each mini-column can receive 64 independent binary features (lattice sections are not internally interconnected).

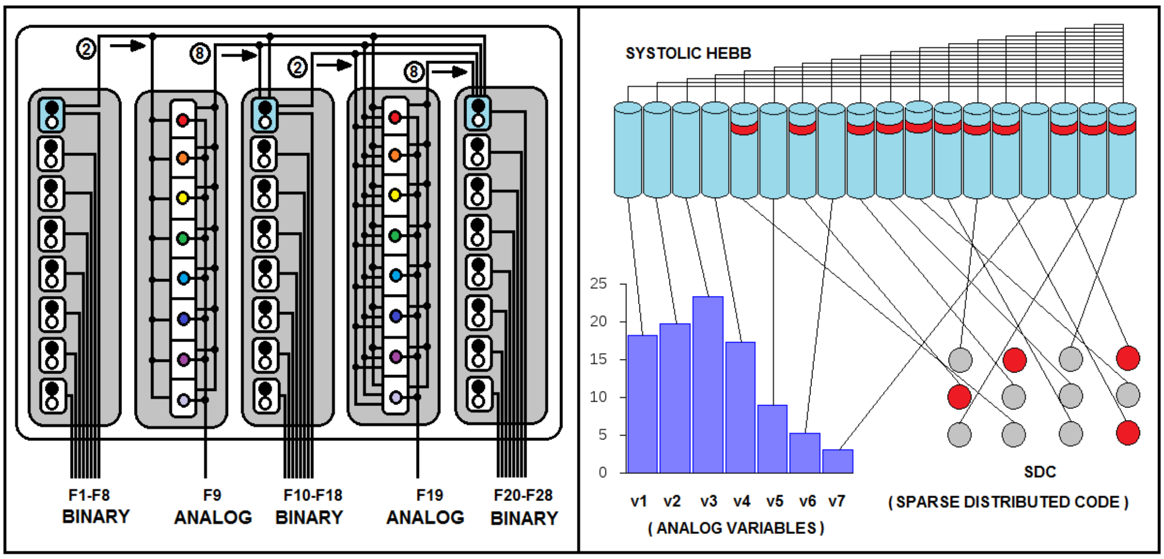


Fig. 21. The picture on the left side shows a hybrid macro-column in which mini-columns processing analog values are interlaced with mini-columns processing binary values. The right side of the picture is a view of the utilization of a hybrid network. A sparse distributed code can be interlaced with analog variables. The rationale, in a machine learning application, is the possibility to link SDCs representing concepts with analog variables, respecting an order that enables the user to split the network into sub-networks, keeping the knowledge of any sub-network intact.

Analog and binary mini-columns can be mixed in hybrid macro-columns (Fig. 21) that process patterns composed of SDCs interlaced with analog values. Hybrid macro-columns require the following modification of matrix (4).

$$\varphi_{m,m'} = \begin{bmatrix} \omega_{1,1} & \omega_{1,2} & \cdots & \omega_{1,n[m]} \\ \omega_{2,1} & \omega_{2,2} & \cdots & \omega_{2,n[m]} \\ \cdots & \cdots & \cdots & \cdots \\ \omega_{n[m'],1} & \omega_{n[m'],2} & \cdots & \omega_{n[m'],n[m]} \end{bmatrix} \qquad (15)$$

n = the number of values representing the features

Matrix (15) is generalized for a network in which each feature can have a different level of discretization.

Methods for introducing variants of the network with synaptic decay and statistical consistency (not always desirable in agents interacting with the environment in the time domain) are currently under investigation and are not presented in this work.

V. Large Delay Synapses and Spatiotemporal Entangled Memory

Synapses with larger delay (multiple of the time required to process a macro-column) have been added to the model to build a relationship in the time domain between input patterns. This is an optional feature that is not mandatory in the SHARP model. The neural network learns the relation between the current values of the features and the past values of the features. With this "transition learning", the noisy input patterns can be recognized through the memory of the delayed synapses, which carry the relationships between the feature values in the time domain (Fig. 22). Spatiotemporal entangled memory (STEM) is a sort of episodic memory entangled with the space of the features: it is our memory of specific parts of past stimuli, from which we can reconstruct other parts of the current stimulus (Fig. 23). A new matrix of forward excitatory con-

Email: luca.marchese@synaptics.org web: www.synaptics.org

nections is introduced to simplify the software implementation. The complete matrix of forward excitatory connections (6) is transformed in (16).

$$\phi = \begin{bmatrix} 0 & \begin{bmatrix} \omega_{1,1} & \left[\rho_1 \quad [\Delta_0\Delta_1...\Delta_\tau]_2 \quad ... \quad \rho_j\right]_{1,2} & ... & \omega_{1,n} \\ \omega_{2,1} & \omega_{2,2} & ... & \omega_{2,n} \\ ... & ... & ... & ... \\ \omega_{n,1} & \omega_{n,2} & ... & \omega_{n,n} \end{bmatrix}_{1,2} & ... & \varphi_{1,(m-1)} & \varphi_{1,m} \\ 0 & 0 & ... & \varphi_{2,(m-1)} & \varphi_{2,m} \\ ... & ... & 0 & ... & ... \\ 0 & 0 & ... & 0 & \varphi_{(m-1),m} \\ 0 & 0 & ... & 0 & 0 \end{bmatrix} \tag{16}$$

$\Delta_x =$ synapse with delay x

From (10) we can obtain the new learning behavior, as described in (17).

$$\forall(0 \le t \le \tau)\forall(1 \le x < N)\forall(y > x)\forall\left(\left(V_{ref(x)}, V_{ref(y)}\right) \ni \left(\left|V_{ref(x)} - V_{inp(x,0)}\right| \le LSUP_x\right) \wedge \left(\left|V_{ref(y)} - V_{inp(y,t)}\right| \le LSUP_y\right)\right) \tag{17}$$

$$\Delta_t \in \rho_j \in \omega_{V_{ref(x)},V_{ref(y)}} \in \varphi_{x,y} =$$

$$\max\left(\begin{array}{l} \left(\Delta_t \in \rho_j \in \omega_{V_{ref(x)},V_{ref(y)}} \in \varphi_{x,y}\right), \\ \left(\left(\left((\Delta_t)_{MAX} \in \rho_{MAX}\right) - \left(\dfrac{(\Delta_t)_{MAX} \in \rho_{MAX}}{V_{MAX}}\right) \times \max\left(\left|V_{ref(x)} - V_{inp(x,0)}\right|, \left|V_{ref(y)} - V_{inp(y,t)}\right|\right) \times \theta\right)\right) \end{array} \right)$$

where: $\theta \le 1 =$ constant (delay ranges distribution)

$$\rho_{MAX} = \left\{(\Delta_0)_{MAX}, (\Delta_1)_{MAX}, ..., (\Delta_\tau)_{MAX}\right\} \ni \left((\Delta_n)_{MAX} < (\Delta_{n-1})_{MAX} \times \left(1 - \frac{LSUP_{MAX}}{V_{MAX}}\right) \times \theta\right)$$

The new recognition law from (7) and (8) is described in (18) and (19).

Email: luca.marchese@synaptics.org web: www.synaptics.org

$$
\begin{aligned}
&\forall j \forall t \forall x \left(M_{j,x,t} = \varnothing \right); \forall j P_j = \varnothing; \\
&\forall j \forall t \forall x \left(\forall y \neq x \left(\Delta_t \in \rho_j \in \omega_{inp(x,0),inp(y,t)} \in \varphi_{x,y} \right) = 0 \Rightarrow M_{j,x,t} = M_{j,x,t} \cup \{ y \} \right) \\
&\forall j \forall x \left(M_{jx} = \bigcap_{t=0}^{\tau} M_{j,x,t} \right) \\
&\text{given} \quad f : x \rightarrow \# M_{jx} \quad \forall j \forall i \leq LNUM \left(\left(\# P_j < LNUM \right) \Rightarrow P_j = P_j \cup \left\{ \arg\max_{x=1, x \notin P_j}^{N} \left(f(x) \right) \right\} \right) \\
&\forall j \left(\left(\# P_j > LNUM \right) \Rightarrow \left(\chi_j = 0; L = L \cup \{ j \} \right) \right)
\end{aligned}
\tag{18}
$$

We have to compute a fuzzy-OR operation between equivalent connections with different delays (19). The fuzzy-OR ensures that synapses with lower delays win over synapses with higher delays.

$$
\forall j \chi_j = \frac{\min_{x=1}^{N-1} \left(\min_{y=x+1}^{N} \left(\max_{t=0}^{\tau} \left(\Delta_t \in \rho_j \in \omega_{inp(x),inp(y)} \in \varphi_{x \notin P_j, y} \right) \right) \right)}{\# P_j + 1} \tag{19}
$$

An issue related to the use of multiple delayed synapses is the growth of crosstalk between the learned patterns. In the limited scope of the tests performed by the author, two or three delays (τ1, τ2, τ3) have been used (Fig. 24). The tests have examined the continuous recognition of a flow of time-correlated patterns (i.e., features extracted from the frames of a movie) with the addition of noise to the features extracted from the frames. The use of delayed synapses implements a type of distributed storage called STE-SDS (spatio-temporal entangled sparse distributed storage) [17] and is useful only within a context of time-correlated patterns.

Email: luca.marchese@synaptics.org web: www.synaptics.org

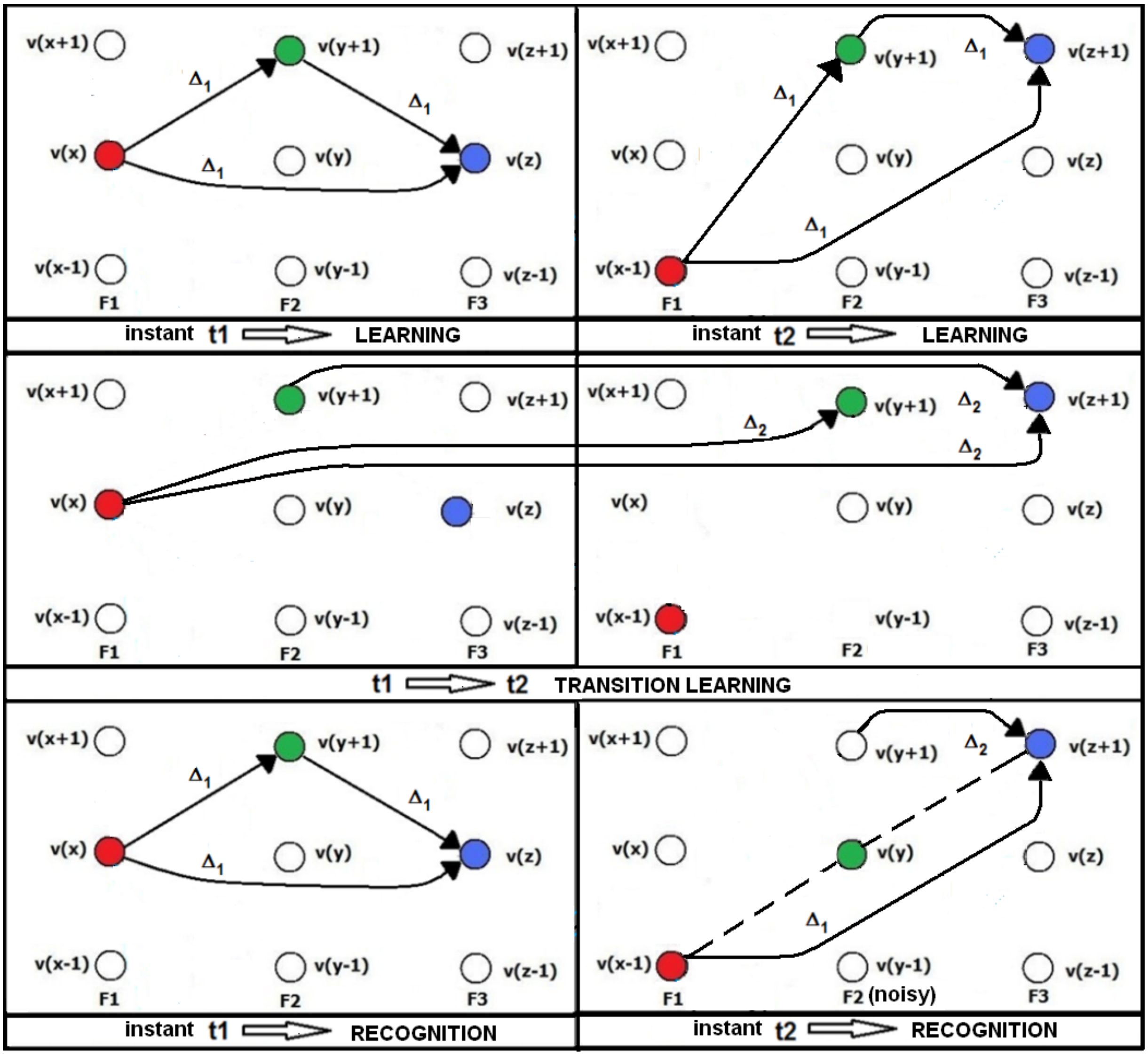


Fig. 22. (TOP) The learning process in the following two instants is performed. At instant t1, synapses Δ1 are modulated. At instant t2, synapses Δ1 are modulated. (MIDDLE) The learning process in the transition between the two following instants is performed. Synapses Δ2 are modulated between the resonating neurons at instant t1 and the resonating neurons at instant t2. (BOTTOM) On the left side, the same learned pattern is recognized using Δ1 synapses. On the right side, the presented pattern has one noisy feature and the pattern is recognized thanks to the memory of the previous pattern using one Δ2 synapse.

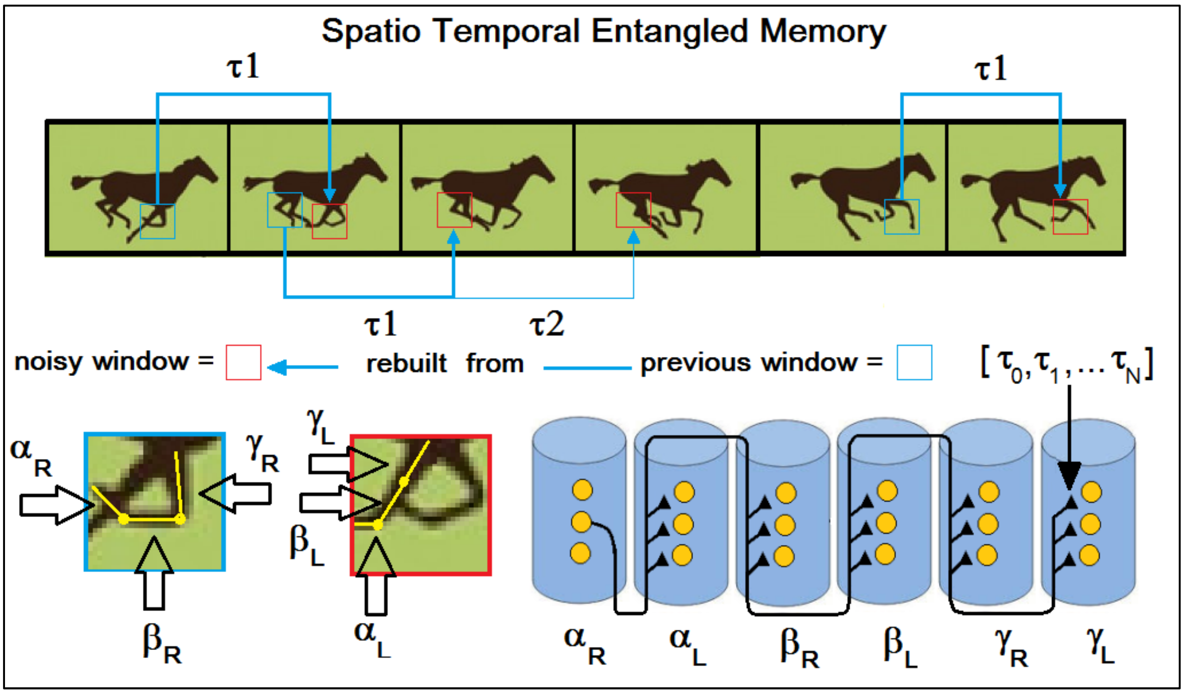


Fig. 23. The picture is a visual representation of the concept of Spatiotemporal Entangled Memory.

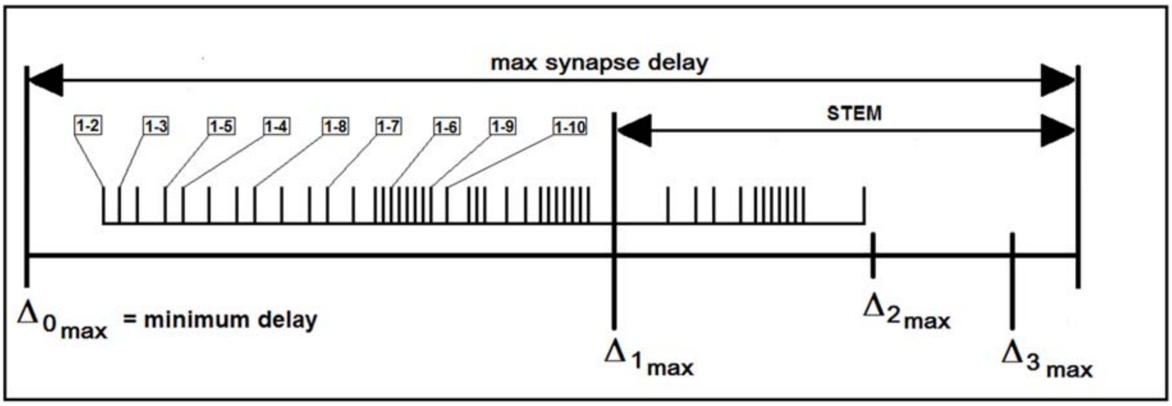


Fig. 24. The picture shows the spikes raster of a systolic path. The total time is the period required to complete the processing of a cortical macro-column and is divided in two basic ranges associated with the current stimulus and the STEM. The post-synaptic timings of the spike emitted by the resonant neuron in the first mini-column are evidenced. The macro-column of this test has 10 mini-columns.

V. Weighted Rule Extraction

The rules can be extracted during the learning process from the dataset using the neural network as a filter, or directly from the synapses of the neural network.
During the learning process, the redundant rules will be automatically discarded through a recognition process preceding any learning process. The conflicting rules, generated by noisy or corrupted data in the training set, can be detected and discarded.
An example of a record in the training set for the SHARP neural network is

$$VAL\_REF(1), LSUP(1), VAL\_REF(2), LSUP(2), \ldots, VAL\_REF(N-1), LSUP(N-1), VAL\_REF(N), LSUP(N), CLASS$$

The extracted rule is:

```
IF((VAL_REF(1) - LSUP(1) ≤ VAL_INP(1) ≤ VAL_REF(1) + LSUP(1)) AND (VAL_REF(2) - LSUP(2) ≤
VAL_INP(2) ≤ VAL_REF(2) + LSUP(2)) AND … AND (VAL_REF(N-1) - LSUP(N-1) ≤ VAL_INP(N-1) ≤ VAL_REF(N-1) +
LSUP(N-1)) AND (VAL_REF(N) - LSUP(N) ≤ VAL_INP(N) ≤  VAL_REF(N) + LSUP(N)) THEN
 [CLASSIFICATION = CLASS,
  STRENGTH = (MAX_STRENGTH / MAX_VAL) * MIN((MAX_VAL - |VAL_REF(1) - VAL_INP(1)|),(MAX_VAL -
                                   |VAL_REF(2) - VAL_INP(2)|),…,(MAX_VAL - |VAL_REF(N-1)
                                    - VAL_INP(N-1)|), (MAX_VAL - |VAL_REF(N) - VAL_INP(N)|))]
```

The pattern is inputted to the network prior to the learning process and the decisions are made on the basis of the recognition result, as shown in Table 3.

recognized	desired class	strength > thrsh	strength < Max	action
🟢	🟢	🟢	⚪	example not learned rule discarded
🟢	🟢	🔴	🟢	example learned rule accepted
🟢	🔴	⚪	🟢	example learned rule accepted
🟢	🔴	⚪	🔴	warning
🔴	⚪	⚪	⚪	example learned rule accepted

Table 3. In the table: thrsh = fixed threshold; warning = a model with statistical consistency is required to process the database.

LSUP can be computed automatically by the neural network using (11). The extraction of the rules from the synapses of the neural network when the training set is no longer available requires a longer process. The algorithm must scan the network from the first value of the first mini-column and find all of the connections to all of the subsequent mini-columns. The same process must be repeated for each value and for each subsequent mini-column. The result is a set of (M-1) * V matrices that is processed iteratively as described in Fig. 25. In this paper, the rule extraction process is only outlined, and a more detailed explanation will be the subject of a future work.

Copyright © Luca Marchese. ORC-ID: http://orcid.org/0000-0001-7903-7540 NCAGE: AK845

Email: luca.marchese@synaptics.org web: www.synaptics.org

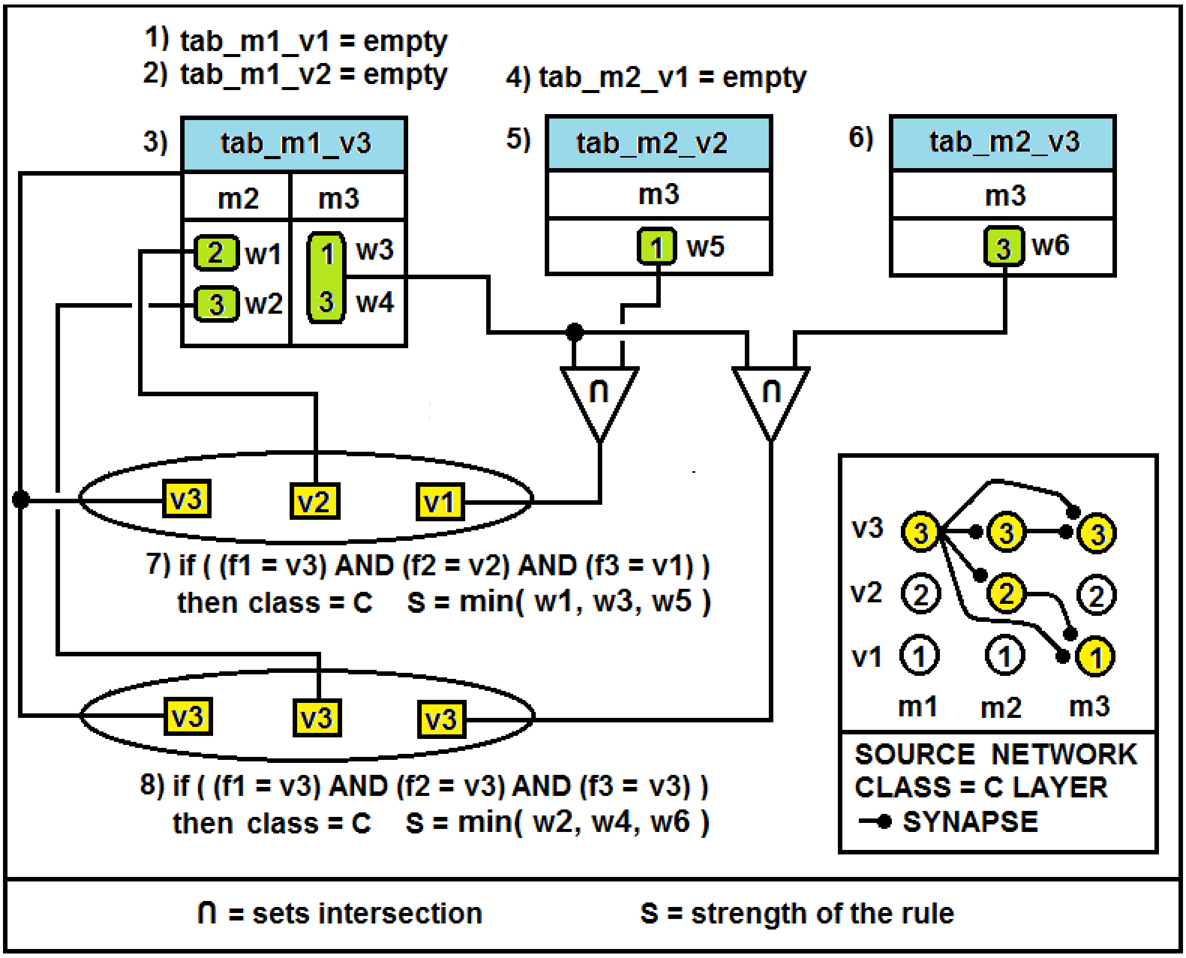


Fig. 25. The picture shows a simple example of the rule extraction process. If M is the number of mini-columns and V is the number of possible values of the features, then the process generates (M-1)*V matrices. The first matrix has M-1 columns, while the last matrix has only one column. The process must be repeated for any macro-column associated with a specific class.

VI. One Step Deep Machine Learning on Von Neumann Computers

SHARP has the following features:

1) Learning and recognition times are independent of the number of learned patterns.
2) Learning and recognition times increase with the number of features, following a quasi-quadratic law.
3) A limited number of categories are managed by a network, in order to limit the memory size and speed-up the process by avoiding iterations.
4) The learning process is realized in a single step.

These characteristics suggest that the network model is well suited for deep hierarchical structures of multiple networks. Indeed, in a hierarchical structure, the features have different levels of detail: the low level features (inputs) can be managed by multiple networks whose outputs are the higher level features for the following stages, and so on. A limited number of categories can be managed using the SDC, as explained in Fig. 26. Therefore, the use of multiple networks allows for the management of a large number of features and a large number of categories with hierarchical learning/recognition processes. The advantage is that the time required for

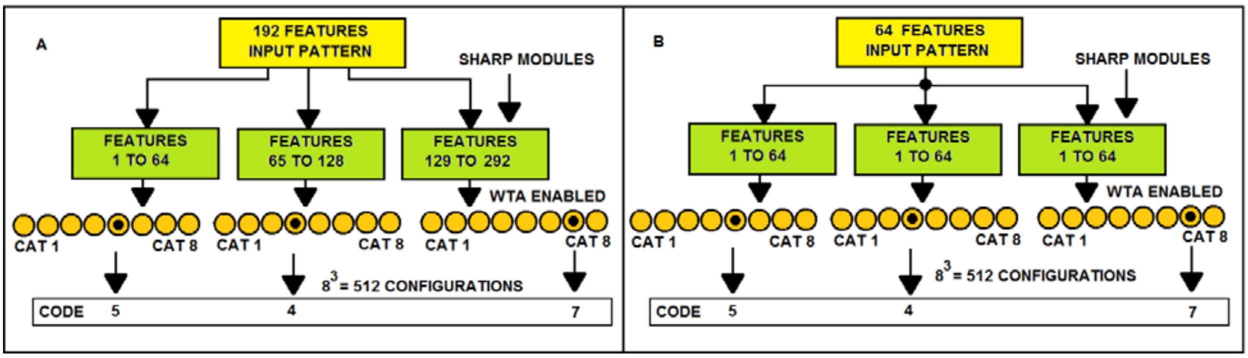


Fig. 26. (A) The picture shows how a large pattern can be recognized using three SHARP modules and sparse distributed code in a single layer. The first token of the pattern is recognized with eight categories by the first SHARP module, the second token by the second SHARP module and so on. (B) The same pattern is the input of three modules trained with different codes in order to produce an SDC. For both configurations, the SDC is represented by the binary codes produced on the category layers (with WTA enabled) of the three modules.

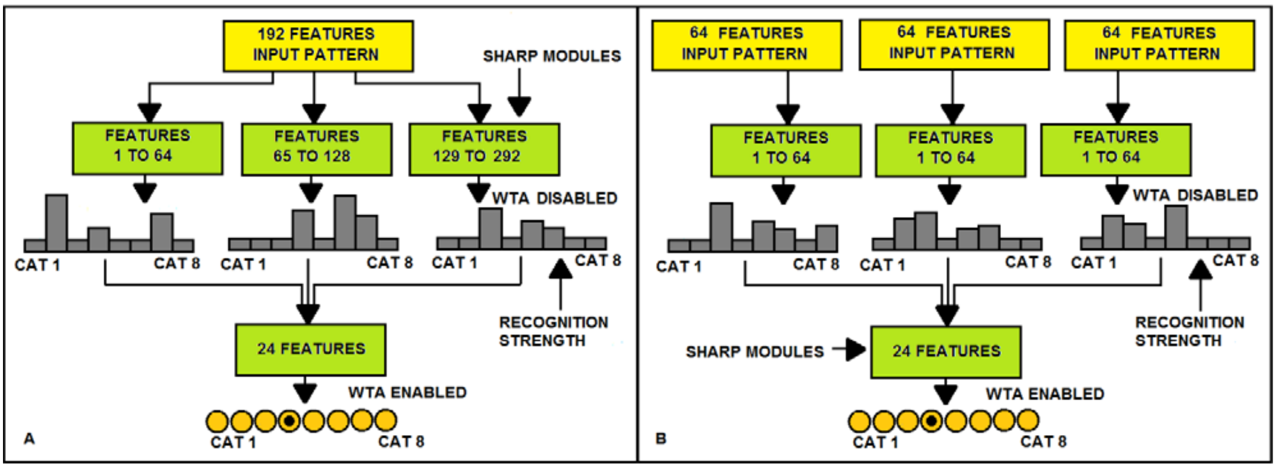


Fig. 27. (A) The picture shows how a large input pattern can be split into three smaller input patterns and recognized using two layers. The output category layer of the first level has WTA disabled and represents the three vectors of the recognition strengths of the three tokens of the input pattern. In this configuration, the learning process should be performed on the entire dataset for each layer. (B) The picture shows three different sources of input patterns that are recognized using two layers. The output category layer of the first level has WTA disabled and represents the three vectors of the recognition strengths of the three input patterns. In this configuration, the learning process should be performed on the entire dataset for any layer.

the serial execution (recognition/learning) of all of these networks is not affected by the number of learned patterns at any level of the hierarchy. The hierarchy can be realized using the output of the networks with the WTA disabled (Fig. 27) and then processing the vector of the recognition strengths associated with any category. The vector of recognition strengths, generated by a specific input pattern, changes when the network learns new patterns. Therefore, in an offline learning application, each layer should be trained separately with the entire dataset. In an online learning environment, in the lower layers (low-level features) the learning event should be triggered with higher frequency than in higher layers (high-level features). When WTA is disabled in the output layers, the network produces patterns that can be processed by the next layers using both LSUP (analog) and LNUM generalization. The drawback of this solution in an online learning environment is the need for correct management of the dynamic evolution of the output vector of the recognition strengths. With the SDC, the hierarchy can be built by enabling WTA in the output layers and using SHARP modules with binary mini-columns. In this case, the generalization is performed using the LNUM. The LNUM parameter is not learned, in contrast to the LSUP, but should be set for each layer before learning. A three-layer hierarchy processing SDCs in the second and third layers is shown Fig. 28. Binary, analog and hybrid SHARP can be used on the same layer of the hierarchy.

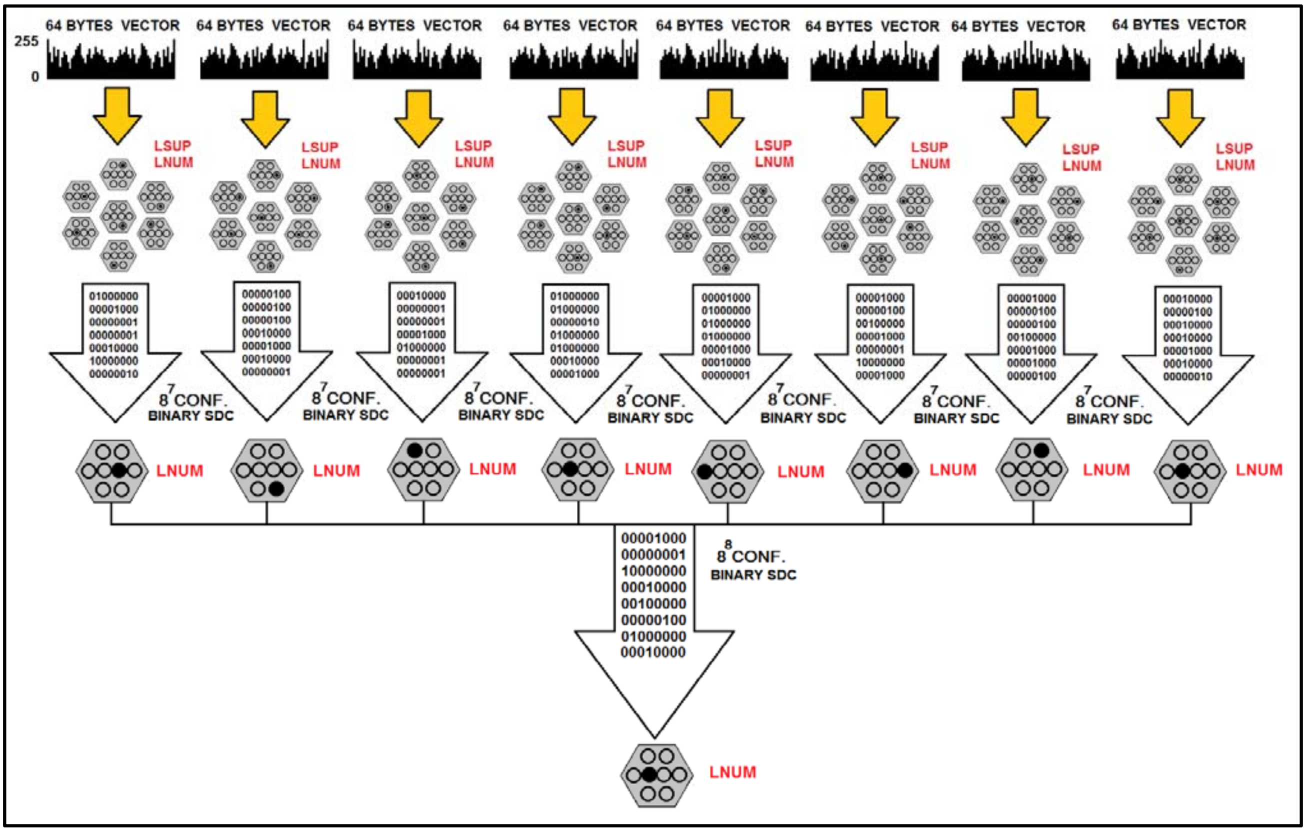


Fig. 28. This is an example of three layers hierarchy. The first layer is composed of 8 clusters of SHARP networks containing 8 macro-columns. Each macro-column in one cluster receives the same 64 bytes input pattern: there are 64 mini-columns in a macro-column. These networks use both LSUP and LNUM generalization. Each cluster is composed of 7 SHARP networks and produces a binary SDC with 8^7 configurations. The 8 binary SDCs generated by these clusters are the input for the second layer composed of 8 SHARP networks with 8 macro-columns and 56 mini-columns. From this layer only the LNUM generalization can be used because the propagated pattern is a binary SDC. The second layer produces an SDC with 8^8 configurations. The third layer is a single SHARP network with 8 macro-columns composed of 64 mini-columns: the output is an 8 categories classification.

V. Questions and Answers

The author has received some feedback and questions related to this work and he wants to publish them together with his answers.

Q. Standard neural networks are not growing with respect to the number of the patterns analyzed in the past, so why is this any difference with respect to other networks?

A. All the neural networks that are based on the concept of prototypes with Influence Field or Vigilance (Adaptive Resonance Theory, Restricted Coulomb Energy, Radial Basis Function) are growing during the learning activity (new prototypes are added). These networks solve the problem of plasticity versus stability and perform a fast learning task based on few cycles. The drawback is that learning and execution on Von Neumann machines require to scan serially the database of prototypes.

Q. Why not a comparison with MLP on a recognition task?

A. The execution of MLP requires a time that is independent by the number of the learned patterns but MLP requires long multiple learning cycles. The comparison is useless because the target of this research is to obtain both learning and execution in a single optimized step (table.1).

Q. The test has been executed using an artificial dataset. Why not a standard dataset like MNIST?

A. The author plans to make tests on MNIST and other standard databases. However, the author is aware that the principal drawback of the network, for pattern recognition applications, is the generalization limited by the Box-Distance. The paper does not want to claim that the SHARP model could outperform other models in pattern recognition tasks. The principal target of the SHARP model is to be a brick for large cognitive networks.

Email: luca.marchese@synaptics.org web: www.synaptics.org

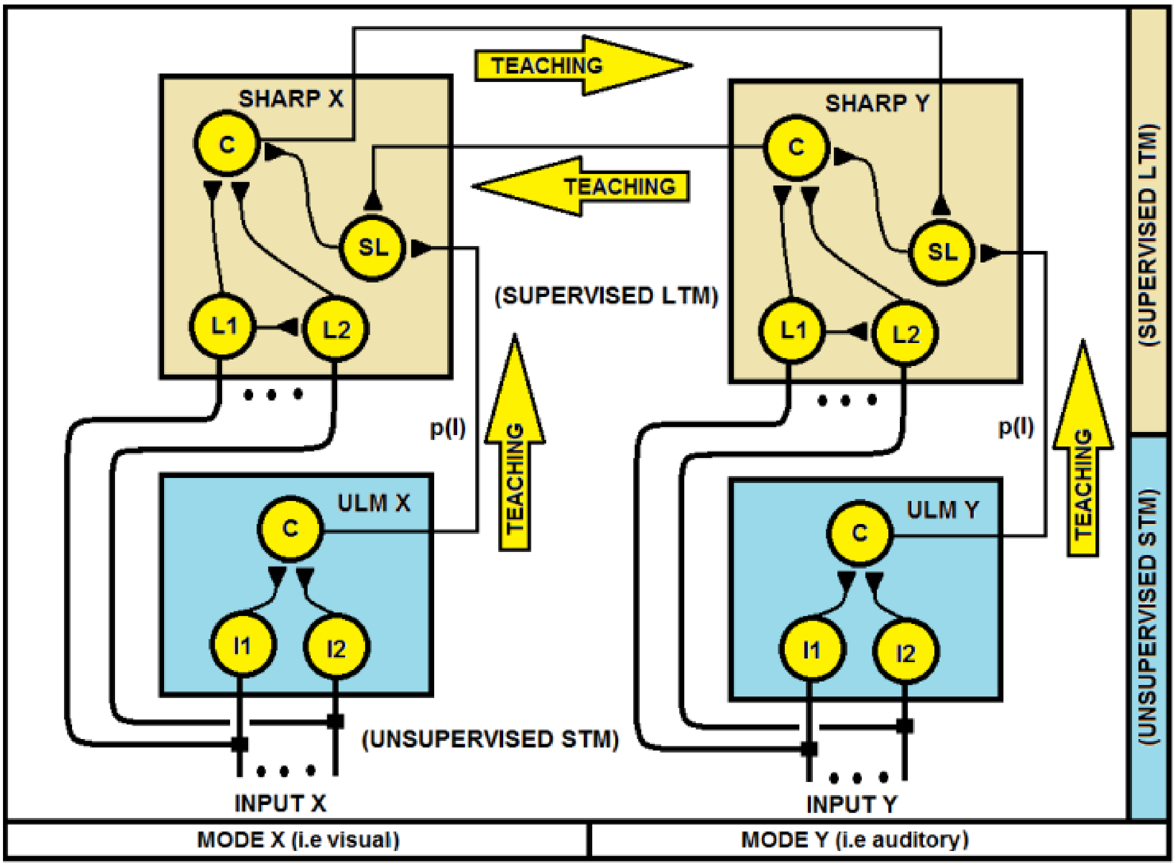


Fig. 29. This is an example of two basic MOSAIC blocks interconnected with a teacher-learner bus between two different learning modalities. ULM modules are small-sized RBF-like neural networks derived from the probabilistic adaptive learning mapper (PALM).

VI. The Conclusions and Future Work

The paper expanded some features of the SHARP neural network model, such as the SDSA property, which enables the network to perform recognition and learning on serial computers in a constant time that is independent of the number of the learned patterns. The author explained the advantages of the SDC in artificial neural systems and supplied references to scientific works related to the biological plausibility of the SDC, with the awareness that the SHARP algorithm and the proposed schemes of connectivity have been designed with the goal of building brain-inspired intelligent systems and that all of the details of the biological plausibility are highly speculative. The paper proposed a model of sparse distributed storage that works in the space of features and the time domain with continuous entanglement between the two dimensions. This property has been called STE-SDS (spatio-temporal entangled sparse distributed storage). The author is working on a framework, called MOSAIC (MOdular Scalability AImed to Cognition), which aims to build large neurocognitive networks in a computer network. The SHARP modules are interconnected with different schemes of hierarchy and in the roles of "teachers" and "learners" [1], building the same target concepts from different streams of sensory inputs (e.g., auditory and vision) (Fig. 29). In the MOSAIC framework, there are unsupervised learning modules (ULM) at every layer of the hierarchy. The role of these modules, derived from the probabilistic adaptive learning mapper (PALM) [18], is to promote frequently occurring patterns from the short-term memory (STM) to the long-term memory (LTM). All of the modules work as asynchronous oscillators that receive analog or SDC input patterns and send/receive SDC patterns. The learning activity is triggered in a specific layer by a ULM or by a learning signal from a SHARP module of another stream. Although the SHARP algorithm is deterministic, the asynchronous communication protocol introduces stochastic behavior into the whole network. A full explanation of the MOSAIC framework will be presented in a future work. The source code (c language) of a simple SHARP NN module can be requested to the author.

REFERENCES

[1] L.Marchese - "SHARP (Systolic Hebb - Agnostic Resonance - Perceptron): A Bio-Inspired Spiking Neural Network Model that can be simulated Very Efficiently on Von Neumann Architectures - American Journal of Intelligent Systems" - 2014; 4(5): 159-195 - SAP (Scientific & Academic Publishing) - doi:10.5923/j.ajis.20140405.01

[2] W. Maass. (1997) – "Networks of spiking neurons: the third generation of neural network models". Neural Networks, 10:1659-1671, 1997.

[3] Gerstein GL, Mandelbrot B (1964) Random walk models for the spike activity of a single neuron. Biophys J 4:41–68

[4] Izhikevich E.M. (2001) – "Resonate-and-Fire Neurons", Neural Networks, 14:883-894

[5] E. M. Izhikevich, "Simple model of spiking neurons," IEEE Trans. Neural Networks, vol. 14, pp. 1569–1572, Nov. 2003.

[6] Carpenter, G.A. & Grossberg, S. (2003), "Adaptive Resonance Theory", In Michael A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, Second Edition (pp. 87-90). Cambridge, MA: MIT Press

[7] Rudin, Walter (1964). Principles of Mathematical Analysis. New York: McGraw-Hill. p. 151.

Email: luca.marchese@synaptics.org web: www.synaptics.org

[8] Broomhead, David H.; Lowe, David (1988). "Multivariable Functional Interpolation and Adaptive Networks". Complex Systems 2: 321–355

[9] Buhmann, Martin D. (2003), “Radial Basis Functions: Theory and Implementations”, Cambridge University Press, ISBN 978-0-521-63338-3.

[10] David A. Patterson and John L. Hennessey – “Computer Organization and Design: the Hardware/Software Interface”, 2nd Edition, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998, p.751

[11] Gerard J. Rinkus - “Quantum Computation via Sparse Distributed Representation” – NeuroQuantology - Vol 10, No 2 (2012)

[12] Gerard J. Rinkus – “A cortical sparse distributed coding model linking mini- and macrocolumn-scale functionality” – frontiers in NEUROANATOMY – 2 June 2010

[13] Rinkus, G., and Lisman, J. (2005). “Timeinvariant recognition of spatiotemporal patterns in a hierarchical cortical model with a caudal-rostral persistence gradient”. In Society for Neuroscience - Annual Meeting, Washington, DC.

[14] Albright, T. D., Desimone, R., and Gross, C. G. (1984). “Columnar organization of directionally selective cells in visual area MT of the macaque”. J.Neurophysiol. 51, 16–31.

[15] P Földiák, 2002, “Sparse coding in the primate cortex” , in The Handbook of Brain Theory and Neural Networks, Second Edition, pp 1064-1068, ed. Michael A. Arbib, MIT Press, ISBN 0-262-01197-2.

[16] Ohki, K., Chung, S., Ch’ng, Y. H., Kara, P., and Reid, R. C. (2005). “Functional imaging with cellular resolution reveals precise micro-architecture in visual cortex”. *Nature* 433, 597–603.

[17] L.Marchese – Advances in Neural Networks – SPRINGER - CHAPTER 153-160: Spatial-Temporal Entangled Sparse Distributed Storage (STE-SDS) and Sparse Distributed Code (SDC) in the Systolic Hebb Agnostic Resonance Perceptron (SHARP) Proposed as Hypothetical Model Linking Mini and Macro-Column Scale Functionality in the Cerebral Cortex.

[18] L.Marchese- “PALM – Probabilistic Adaptive Learning Mapper” - Natural Intelligence – The INNS Magazine VOL.2 ISSUE 1 (2013)

 ORC-ID: http://orcid.org/0000-0001-7903-7540 NCAGE: AK845

Email: luca.marchese@synaptics.org web: www.synaptics.org