THE PYRAMIDAL UNIVERSAL CELL: A NEW APPROACH FOR COMPUTATION IN CELLULAR NEURAL NETWORKS

by

Radu Dogaru, Kenneth R. Crounse, and Leon O. Chua

Memorandum No. UCB/ERL M97/41

3 June 1997

THE PYRAMIDAL UNIVERSAL CELL: A NEW APPROACH FOR COMPUTATION IN CELLULAR NEURAL NETWORKS

by

Radu Dogaru, Kenneth R. Crounse, and Leon O. Chua

Memorandum No. UCB/ERL M97/41

3 June 1997

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

The Pyramidal Universal Cell: A New Approach for Computation in Cellular Neural Networks

Radu Dogaru[1], Kenneth R. Crounse and Leon. O. Chua
University of California Berkeley,
Nonlinear Electronics Laboratory, 258 M Cory Hall,
Berkeley, CA-94720 1770
E-mail: radu_d@fred.eecs.berkeley.edu

June 3, 1997

Abstract

A significant increase in the information processing abilities of CNNs demands powerful information processing at the cell level. In this paper, the defining formula and the main properties of such a cell are presented. Since it is able to implement any Boolean function, its functionality expands on those of digital RAMs by adding new capabilities such as learning and interpolation. While it is able to embed all previously accumulated knowledge regarding useful binary information processing tasks performed by standard CNNs, the pyramidal universal cell provides a broader context for defining other useful processing tasks, including extended grey scale or colour image processing as well. Examples of applications in image processing are provided in this paper. Implementation issues are also considered. Assuming some compromise between area and speed, a VLSI implementation of CNNs based on pyramidal cells offers a speed-up of a million times when compared with corresponding software implementations.

1. Introduction.

The CNN paradigm and its implementation via the CNN universal machine [2] has already proved very useful for a wide range of information processing tasks. The main element which influences these processing capabilities is the basic cell, which collects information from its neighbours and provides a synthesis of this information in the form of a single output. Such cells may operate in feed-forward or feed-back layers. Various cell models have been described in the literature [2], each having advantages and drawbacks. In our search for a universal cell, this paper proposes a new approach, inspired by both the theory of radial basis functions [11] and of piece-wise linear approximates [1][10]. It is defined as a static cell, i.e. its time evolution is memoryless. We will also consider a bounded universe of signals (i.e. the domain over which each input and the output may vary) on [-1,1]. The boundaries of this domain correspond to the "false" and "true" Boolean logic truth values. We will also find it convenient to consider a more relaxed logic, somewhat related to the fuzzy logic concepts [13]. We will call it Boolean Interval Logic. According to this logic, "falsity" and "truth" are assigned degrees corresponding to points in the [-1,0) and (0,+1] domains respectively. A third, "don't know" state is introduced to define perfect ambiguity. It will obviously correspond to the value 0. A general formula for the output y of the cell is:

---

[1] On leave from University "Politehnica" of Bucharest with a Fulbright grant (permanent e-mail address: radu_d@atm.neuro.pub.ro and http://atm.neuro.pub.ro/~radu_d).

The work was also partially suported by the Office of Naval Research under grant N00014-96-1-0753.

$y = f(u_1, u_2, .. u_n, w_1, w_2, .. w_m)$ where $u_1, u_2, .. u_n$ are the inputs and $W = \{w_1, w_2, .. w_m\}$ is a set of parameters.

What would be the desired features for such a universal cell ?

(1) It must be able to implement any of the 2^{2^n} possible Boolean functions in Boolean Interval logic.

This means that for each Boolean function, a particular set of parameters can be found such that the input-output relationship will correspond to the truth table of the defined function. For crisp Boolean logic, digital RAMs fulfil this property. However, they are not able to operate with a Boolean Interval logic.

(2) Assuming that feature (1) is fulfilled, the algorithm which generates the parameters starting from a given truth table should be precisely defined, computationally tractable and have guaranteed convergence.

Although various theorems in the neural network literature [8][9] claimed that polynomial or feed-forward layered structures are universal approximators and thus they should be able to "learn" parameters corresponding to the implementation of any Boolean Interval logic function, in some cases the specification of the neural net structure (e.g. the number of units in the hidden layer) is not exact and in other cases the convergence of the learning algorithm is not guaranteed. We will prove in this paper that a particular class of radial basis function networks is able to fulfil this property with a zero-complexity algorithm for determining its parameters.

(3) The ability to store knowledge by learning from examples even when a complete truth table is not available. The convergence of the learning process should also be guaranteed.
This situation may occur in numerous practical situations. For example, in biological experiments one may use multiple spatially placed stimuli but will be not able to provide the whole range of Boolean Interval combinations. However the output information implied by these measurements is more than nothing and it is of practical interest to store it in the form of a partial or incomplete model by such a universal cell. Since most of the practical cases deal with non-linear separable patterns, in order to guarantee the convergence of the learning algorithm, non-linear pre-processing with dimensionality expansion should be performed on the input data [3].

(4) It should be informationally optimal, i.e. the average information I_W required for specifying the parameters should be equal to the information I_{Bf} needed to specify any particular Boolean function. namely $I_{Bf}(n) = 2^n$. Here, averaging was considered over the whole set of parameters corresponding to the implementation of all Boolean functions.

(5) It should have low computational complexity. This means that any function f may be expressed as a composition of a small number of "computational atoms" which are themselves easy to implement in a given technology. An interesting and very efficient solution in this sense was proposed in [4]. However, it has no learning abilities and it was designed to cover Boolean logic only.

As it is shown in section 3, these properties are fulfilled by a particular case of radial basis function network which may be also considered as a PWL (piece-wise linear) nonlinear cell. We called this cell a Pyramidal Universal Cell. In section 2, the defining equation and the proof of its universality with respect to the Boolean logic are presented.

When used within the CNN paradigm [2], the Pyramidal Universal Cell, or simply P-cell has its inputs associated with all neighbours within the prescribed sphere of influence in the CNN grid. In this paper we will discuss 3x3 neighbourhoods i.e. 9 inputs per cell. Due to its universality, interpolation and learning abilities, it is expected that such cells will offer a significant increase in the number of useful processing tasks in CNN systems. For many linear or non-linear cloning templates used in binary image processing tasks [4], one may simply extract the defining truth table ("gene"), allowing the P-cell to store in a unified way all previous knowledge about such useful information processing tasks. Moreover, by exploiting its

interpolation abilities it is expected to expand in a natural way some tasks (e.g. edge detection) from binary to grey-scale processing. But the most important feature is that we can exploit its learning capability to find new applications. One such example in adaptive image restoration is given in Section 4.

As long as the cell model is specified by a composition of only three simple operators (summation/subtraction, absolute value, and scaling by 0.5) it should have a simple VLSI implementation, a very important feature from the perspective of using it in massively parallel information processing systems such as the CNN. In Section 5 several suggestions for implementation are presented and it is shown that such a cell may be implemented with a complexity comparable to that of linear templates if one accepts a speed reduction of two orders of magnitude. However, the overall CNN processing speed is still high enough to ensure real-time image processing (about 6 million times faster than if implemented on a standard microprocessor).

2. The pyramidal basis cell

Consider the following table form (truth table) for specifying any Boolean function with n inputs:

Table 1. Boolean (Interval) truth table

u_1 =	-1	+1	.....	$a_{j,1}$	.....	+1
u_2 =	-1	-1	.....	$a_{j,2}$		
:	:		.....	:	.....	+1
u_i =	-1	-1	.....	$a_{j,i}$	.....	+1
:	:	:	.....	:	.....	+1
u_n =	-1	-1	.....	$a_{j,n}$	.....	+1
"gene"	g_1	g_2		g_j		g_{2^n}

There are 2^n columns in this table. Each column represents a fixed n-bit binary number. Only the bottom row (shaded) contains information which specifies a particular function. In a Boolean truth table $g_j \in \{-1,1\}$. However, we will extend this table for using it with any Boolean Interval logic value; namely $g_j \in [-1,1]$. Since the bottom row contains the complete information which defines the behaviour of the cell, we will call the vector $\mathbf{g} = [g_1, g_2, .. g_j, .. g_{2^n}]$ a "gene"[2].

Theorem 1: For any binary combination of inputs $(u_1, u_2, ..., u_n)$, $u_i \in \{-1,1\}$, the class of functions defined by :

$$y = \sum_{j=1}^{2^n} g_j \otimes_\alpha P_j(u_1, .., u_i, .. u_n) \qquad (1.1)$$

[2] The "gene" is a short name proposed in [15] and it designates the minimal truth table associated with a specific Boolean function. A minimal truth table is a ordered 2^n binary vector which defines the variable part of the truth table. For P-cells, the components of this vector may vary within the [-1,1] domain. A convenient graphical representation for a binary "gene" was proposed in [4] for $n=9$ as an "image" having 16x32 pixels, where each pixel is associated with one particular component of the vector, and is assigned a brightness equal to its value. The upper left-most pixel corresponds to the first entry (all "n" inputs are -1) in the truth table and all other pixels follow in the normal writing order.

where:
$$P_j(u_1,..,u_i,..u_n) = \mathrm{r}\left(1-\left(\frac{1}{2^\gamma}\sum_{i=1}^{n}\left|u_i - a_{j,i}\right|^\gamma\right)^\beta\right), \qquad (1.2)^3$$

$$\beta > 0; \gamma > 0 \text{ are real numbers}$$

and[4]: $$x \otimes_\alpha y = \left|\frac{x+y}{2}\right|^\alpha - \left|\frac{x-y}{2}\right|^\alpha, \alpha \in \{1,2\} \qquad (1.3)$$

is equivalent to the corresponding Boolean Interval logic function specified in Table 1. Moreover, for $n \geq 2^\gamma$ if all inputs are ambiguous (0) the output is ambiguous (0) too.

In what follows we will call the cell defined by (1.1)-(1.3) a pyramidal cell. It is a particular case of a radial basis network [11] when the basis function is pyramidal, first introduced and investigated in [7]. When $\gamma = \beta = 1$ the equation (1.2) defines a hyper-pyramid in an $(n+1)$-dimensional space. For $n=2$ its geometrical representation is a square-base pyramid with its base lying in the input space, and having diagonals of length 4 (See Fig.1 for a contour plot projection, j=1). Much more complex geometrical figures may be obtained by choosing different values for the real-valued β, γ parameters.

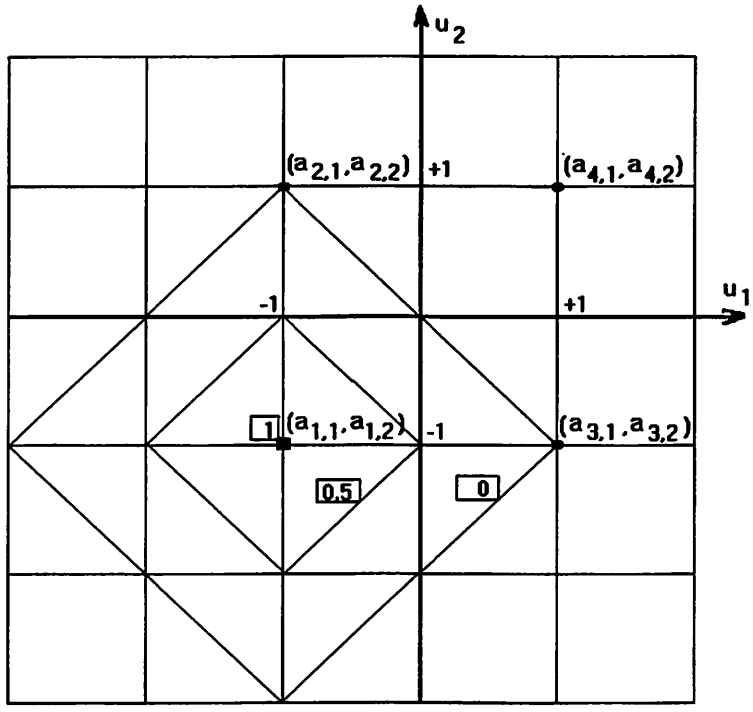

Fig.1. The contour plot representation for a pyramidal basis function corresponding to j=1. The number within a square denotes the value of P_j corresponding to the contour (level curve) nearby; $a_{j,i}$ denotes the fixed binary entries in the corresponding truth table of a Boolean Interval logic function to be implemented with pyramidal basis functions.

---

[3] $\mathrm{r}(x) = \frac{|x|+x}{2}$ is the "standard half-wave rectification" function which has a simple analog implementation: it is just a "concave resistor" [14, p.78] with $(G,E)=(1,0)$.

[4] For $\alpha = 1$ this operator is called a "comparative synapse" and its properties are described in [6]. For $\alpha = 2$ it reduces to an ordinary multiplication $x \otimes_2 y = xy$.

Example (n=2, $\alpha = \beta = \gamma = 1$): Implementing the XOR function.

The complete expanded form of eq (1.1) in this case is:

$$\begin{aligned} y = \{ & -\left|2-\left|u_1+1\right|-\left|u_2+1\right|\right|-\left(2-\left|u_1+1\right|-\left|u_2+1\right|\right)+ \\ & +\left|2-\left|u_1-1\right|-\left|u_2+1\right|\right|+\left(2-\left|u_1-1\right|-\left|u_2+1\right|\right)+ \\ & +\left|2-\left|u_1+1\right|-\left|u_2-1\right|\right|+\left(2-\left|u_1+1\right|-\left|u_2-1\right|\right)- \\ & -\left|2-\left|u_1-1\right|-\left|u_2-1\right|\right|-\left(2-\left|u_1-1\right|-\left|u_2-1\right|\right) \} / 4. \end{aligned} \tag{1.5}$$

It may be further be reduced to the following canonical representation of a piece-wise linear function:

$$y=\left\{-\left|2-\left|u_1+1\right|-\left|u_2+1\right|\right|+\left|2-\left|u_1-1\right|-\left|u_2+1\right|\right|+\left|2-\left|u_1+1\right|-\left|u_2-1\right|\right|-\left|2-\left|u_1-1\right|-\left|u_2-1\right|\right|\right\} / 4 \tag{1.6}$$

Fig 2. represents the output of a pyramidal cell implementing the Boolean XOR function for binary inputs. From the contour plot representation in Fig.2.(a) it follows that this logic function is also implemented in the context of Boolean Interval logic where not the value but rather the sign define the membership to one of the "false", "true" or "ambiguous" categories for both inputs and output. The spatial representation in Fig.2.(b) shows how this function was obtained as a combination of pyramids.

Proof (Theorem 1): The proof consists in proving the following sequence of propositions:

P.I. For any particular input vector $\mathbf{u}^k = \left[a_{k,1},..a_{k,i},..a_{k,n}\right]$ $P_j(\mathbf{u}^k) = \begin{cases} 1 \text{ if } k = j \\ 0 \text{ if } k \neq j \end{cases}$

P.II. For $n \geq 2^\gamma$ if all inputs are ambiguous (0) $P_j(\mathbf{0}) = 0 \ \forall j$.

P.III. $y(\mathbf{u}^k) = g_k$ and $y(\mathbf{0}) = 0$ (The Binary Interval logic truth table definition)

Proof (P.I.): If $k = j$, all absolute value terms in (1.2) become 0, and thus $P_j(u^j) = 1$. The geometrical interpretation is that the input vectors lies exactly on the projection of the pyramid's apex.

If follows froma property of the matrix $\left\{a_{j,i}\right\}_{j=1,2^n;i=1,n}$ (the fixed part of the truth table) that when $k \neq j$, at least one of the absolute value terms in (1.2) is positive. Moreover, since both $u_{k,i}$ and $a_{j,i} \in \{-1,1\}$ this positive value is 2.

Hence: $\left(\frac{1}{2^\gamma}\sum_{i=1}^{n}\left|u^k{}_i - a_{j,i}\right|^\gamma\right)^\beta \geq 1$. It follows immediately that $P_j(u^k) = 0$. The geometrical interpretation is that the input vector now lies on the basal edges of the pyramid, or outside of it.

For binary inputs, each pyramidal basis function performs the same logic function as a "minterm" in the digital implementation of Boolean functions. The gene of a minterm Boolean logic function has only one element +1 and all the others -1. However, in contrast to logic gates, the pyramidal basis functions can also interpolate between the crisp, binary input values.

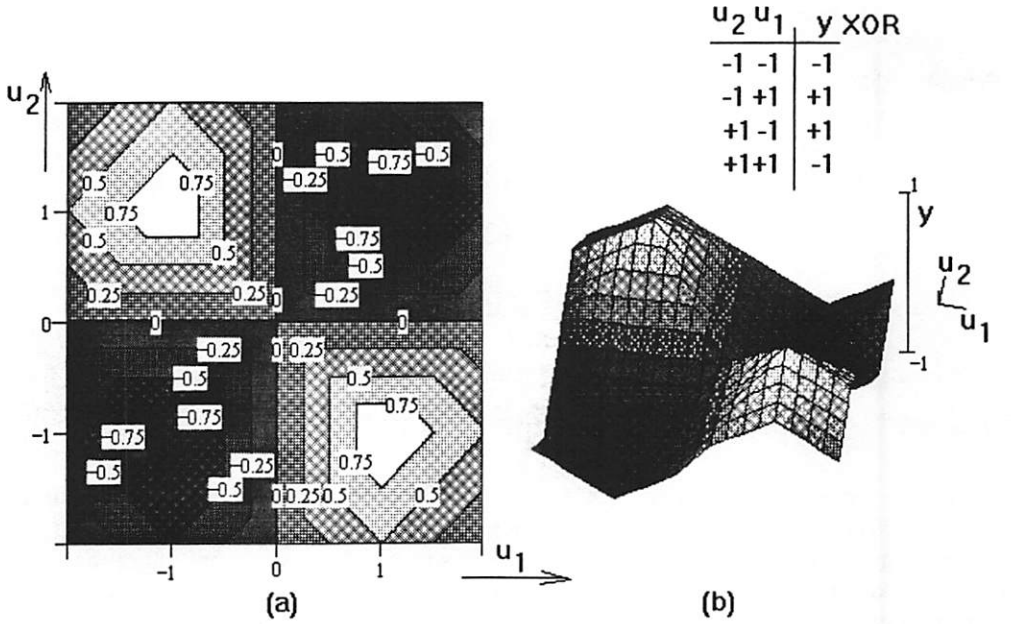

Fig.2: Pyramidal cell implementing the Boolean Interval logic XOR function (black = -1, white = +1): (a) The truth logic which defines the Boolean XOR function is preserved with respect to the sign of both inputs and output; (b) The spatial representation shows the output as a combination of pyramids.

Proof (P.II): Since $u_i^k = 0 \ \forall k,i$ and $|a_{j,i}| = 1$ it follows that $\left(\frac{1}{2^\gamma}\sum_{i=1}^{n}\left|u^k{}_i - a_{j,i}\right|^\gamma\right)^\beta = \left(\frac{n}{2^\gamma}\right)^\beta$. For $n \geq 2^\gamma$ this term is always >1 and thus $P_j(\mathbf{0}) = 0 \ \forall j$.

Proof (P.III): This follows immediately from (1.1) as a consequence of P.I. and P.II and based on the following property: If $\alpha \in \{1,2\}$ and $|x| \leq 1$ then $x \otimes_\alpha 1 = x$ and $x \otimes_\alpha 0 = 0$, which was proved in [6]. This complete our proof of Theorem 1.

3. Properties of the pyramidal cell

In what follows we will briefly show that all properties mentioned in the introduction are fulfilled by the pyramidal cell described by (1.1)-(1.3).

P1. Is a direct consequence of the Theorem 1. For $\gamma = 1$ any cell having at least two inputs will be also able to process Binary Interval logic and provide uncertain output when inputs are uncertain. A detailed study on the relationship between parameters and the degree of truth at the output will be presented in a future work. However, for $n=2$ one can easily see from Fig.2(a) that the XOR function considered in Example 1 is implemented not only with respect to the Boolean logic but to the Binary Interval logic too. Fig.3. presents the case of an ambiguously defined function. Its corresponding gene (having 0 elements) may be the result of a learning process with a limited set of stimuli. However, by using further thresholding with a variable parameter this ambiguity can be removed. The equation which describes the new output is $o = \mathrm{sgn}(y - \Theta)$. With the same implementation of y, two Boolean logic functions may be implemented by simply changing the value of the threshold Θ. In our example, by chosing $\Theta = 0.5$, the output o corresponds to the Boolean XOR function. Compared with the implementation given by (1.5), instead of four, only two pyramidal basis are now required. Using this technique makes possible efficient implementation for all Boolean logic functions with a low number of -1 or +1 bits in their corresponding

genes. The procedure consists in rewriting the original gene (which is defined only by -1 and 1 bits) in Boolean Interval logic with ambiguous values. This consists in replacing with 0 the bits in the original gene having the biggest count and then selecting a threshold with the absolute value 0.5 and the same sign as of the remaining non-zero bits. Such a typical example is the minterm Boolean function which has only one bit +1 and all the rest -1. In this case, all -1 bits in the original gene will be replaced by 0 and a threshold having the value 0.5 will be considered. In this case, only one pyramidal basis function is needed to implement the Boolean minterms.

P2. The only parameters of the formula (1.1)-(1.3) which determine the implemented Boolean function are g_j, and they can be copied directly from the "gene" without any additional computation. This is not a trivial property since even for linear threshold gates there is no direct, analytical solution and learning algorithms must be employed to find the set of parameters which allow one to implement a restricted set of Boolean functions. The choice of the three real parameters α, β, γ has no influence as long as the inputs are restricted to a binary state. However, they will become important when Boolean Interval logic is considered for the inputs.

P3. It is a consequence of Proposition P.I. in the above theorem. Indeed, the set of 2^n pyramidal basis functions may be viewed as a non-linear pre-processor which expands a particular n-dimensional binary input vector into a 2^n-dimensional one. As a consequence of P.I. for any binary input vector, the binary vector in this expanded space has only a "1" and the rest of components "0". In a geometrical interpretation, this corresponds to one of the 2^n vertices of an n-dimensional hypercube. There is always an infinite set of hyperplanes which can separate one vertex from the rest. Moreover, the hypercube has a huge set of points (e.g. each edge) which can be separated by hyperplanes and these points corresponds to different Boolean Interval values at the inputs. Equation (1.1) may be thus viewed as an Adaline [12] and trained (parameters g_j are updated according with an error signal) with the LMS algorithm. The convergence is guaranteed not only for all cases corresponding to binary inputs but also for many others which correspond to different combinations of Boolean Interval inputs. For non-binary inputs, the linear separability in the expanded space is influenced and can be controlled by tuning the α, β, γ parameters.

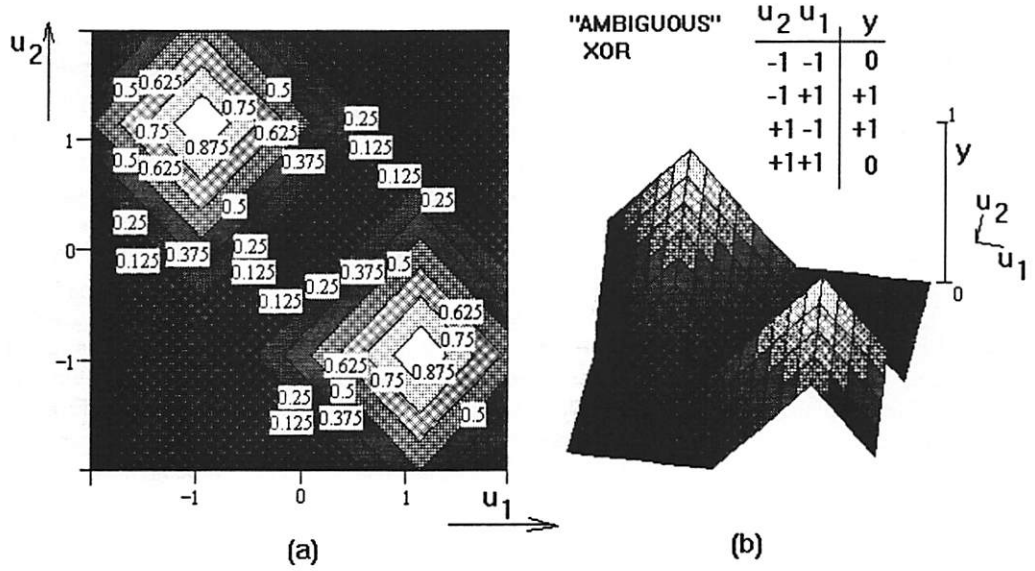

Fig.3: An ambiguously defined function and its implementation with the pyramidal cell (black=0, white=1): (a) Contour plot image; (b) Spatial image

P4. Since the same values g_j which define the truth table are also the parameters of the cell function, $I_W = I_{Bf}$ and thus (1.1)-(1.2) is informationally optimal. This is not a trivial property since for any other known functional which implements Boolean logic functions there are no analytical methods to determine the exact values of both parameters and their representation resolution. However, most of the experimental results cited in the literature as well as our experimental results suggest a ratio $I_W / I_{Bf} \geq 9$ for functionals which partially fulfil P.1. (i.e. only for binary inputs and outputs). The ratio may be lower for functionals, like the Linear Threshold Gates, which however dramatically restrict the number of Boolean functions.

P.5. For the particular case $\alpha = \beta = \gamma = 1$ (1.1),(1.2) and (1.3) can be decomposed into only three species of computational atoms: a) addition/subtraction b) absolute value and c) scaling by a constant 1/2. Observe that there is no ordinary multiplication between coefficients and parameters, as is the case even for the computationally simplest known functionals given by canonical piece-wise linear representations [1]. All of these three atoms can be easily implemented using current technologies. We must however notice that the price paid for all these convenient properties is that the number of basic computational atoms grows exponentially with the number of inputs. However, convenient implementation solutions may be found, as it is shown in Section 5.

4. Exploiting learning in pyramidal universal cells for image restoration

The CNN considered in this section for image restoration tasks, performs the information processing task in the feed-forward layer "B" and is composed of pyramidal universal cells having $\alpha = \beta = \gamma = 1$, which was presented in detail in Section 2. For the experiments described below, the images to be processed are binary, with 32x32 pixels.

The perturbation images have black pixels distributed according to some regular morphology. The goal is to design a CNN cell which allows the corresponding CNN system to be used for removing morphologically known perturbations from any possible input images. For example, we may define the morphology of a particular perturbation as "diagonal lines" and thus the goal will be to design such a CNN cell which allows for removing "diagonal lines" from any possible input image.

In what follows we have considered two such different image perturbations: a) horizontal lines and b) randomly distributed rectangular boxes. For each experiment, the following strategy was considered:

First, a binary random image was artificially generated. This image was then corrupted with the perturbation image. The resulting image was considered as the input while the non-corrupted binary one was considered as "desired response" in a learning process. Our learning scheme consists of applying the Widrow-Hoff (LMS) algorithm at the output layer of the P-cell and thus convergence towards a global optimum is guaranteed (see section 2, P3). In each step of the learning process, one of the CNN's P-cells is excited with a 3x3 pixels window from the input image while an error is computed between its actual output and the desired one associated with the middle pixel of the respective window. Based on this error, the "gene" parameters (which are initially set to 0) are updated. A training epoch consists in sequentially activating all P-cells in the CNN. More epochs may be used until the average training error stabilises. In our experiments, the training lasted four epochs corresponding to 3 minutes of computation on a Pentium 133 processor. The motivation for using a binary noise image as the desired one is that we want the P-cell to learn how to react to as many input combinations as possible. Since regular images provide significantly fewer possible pixel combinations relative to a 3x3 window, the system would not be able to react correctly when a new image is presented (i.e. it will have a low generalization capability).

The results of the learning process may be visualized as "genes" in Fig.4.

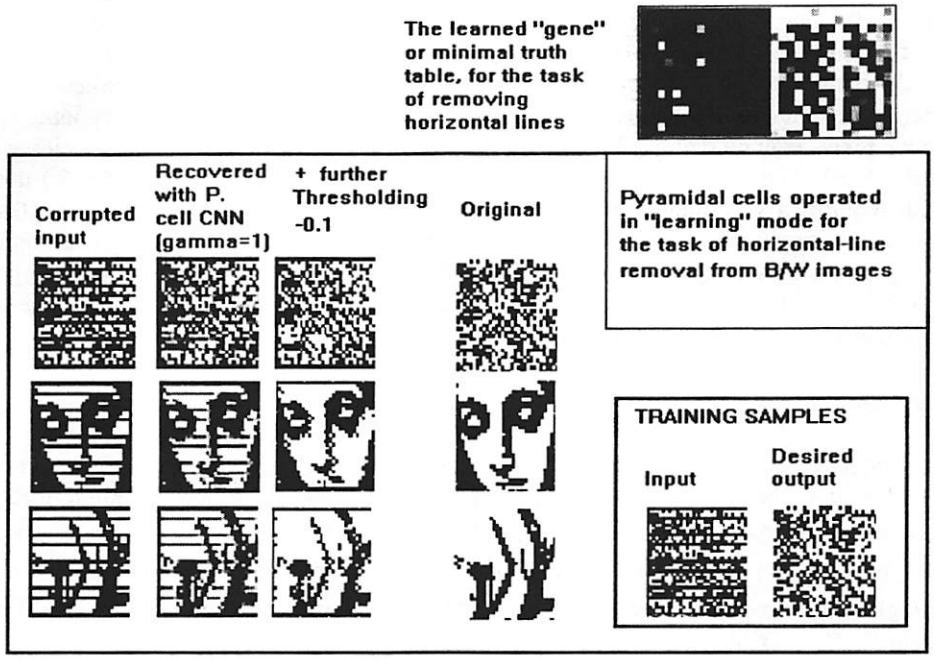

Fig. 4a: Image restoration by learning in CNNs based on pyramidal universal cells: Horizontal-line perturbation

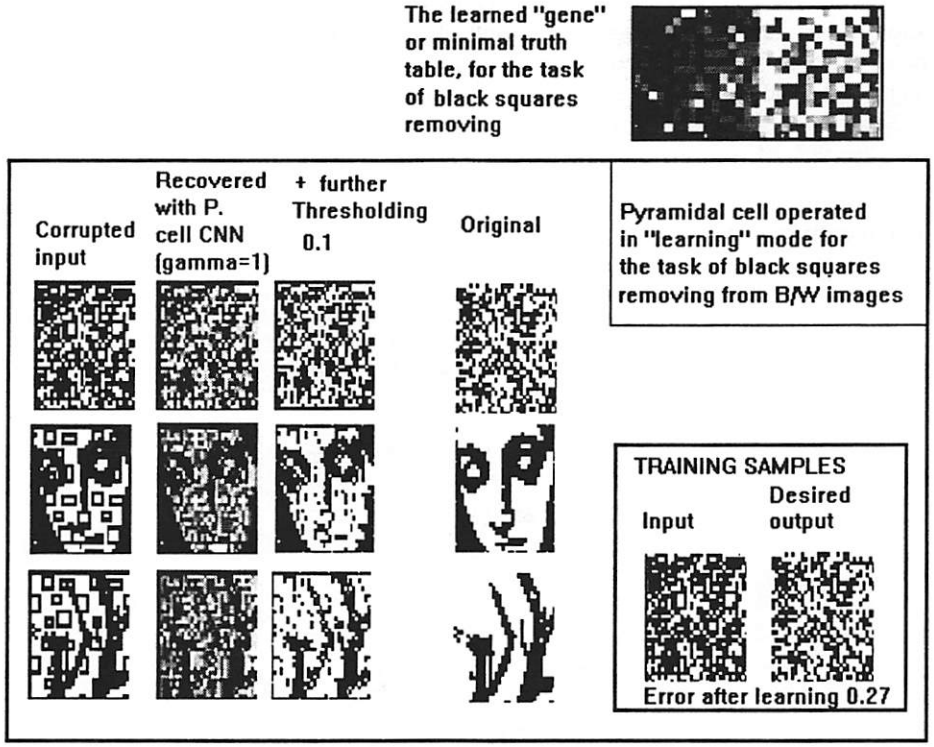

Fig 4.b: Image restoration by learning in CNNs based on pyramidal universal cells: Rectangular-box perturbation.

Observe that some pixels in these non-Boolean genes are "grey" which corresponds to 0 value or "don't know" in the Boolean Interval logic. Such "genes" cannot be defined using other solutions proposed in the literature. In fact, the "grey" pixels in the genes play a significant role in the above image restoration task as it can be inferred from Fig.4. Here, different test images were applied to the CNN with trained P-cells.

Instead of having only two brightness levels (black and white), the processed image also has "grey" pixels (i.e. having 0 values) which may correspond to all input combinations learned as "ambiguous". It also follows that most of these pixels lie in the area corrupted by the perturbation. In other words, the cell learned the perturbation essentially by reacting with the answer "don't know" when some 3x3 window input belonged to a corrupted area. Furthermore, since we are dealing with binary image processing, all ambiguous pixels may be removed, thereby restoring an image which is significantly closer to the original by simply thresholding with a fixed value. In the physical implementation (see Fig.5.) this thresholding operation requires a very simple circuitry corresponding to a simple CNN template. Observe that this strategy can be applied to a large variety of possible regular perturbations, and, as opposed to other methods (e.g. the scratch removal presented in [17]) which need a sequence of operations on the CNN universal machine, the same result is obtained after only one CNN time-step. A detailed analysis of this class of applications will be presented in a future paper.

5. Implementation aspects

The model proposed in Section 2 is well suited for digital hardware or software implementations. Instead of most Radial Basis Functions models, it has the advantage that no multiplication operator is required (when $\alpha = 1$) and it is basically a PWL representation, thus only simple operators such as summation/subtraction and absolute value extraction are required for $\gamma = 1$ (which covers binary image processing). However, the computation time is still large, particularly for large images. For example, to implement a 32x32 CNN using a C++ compiler for the P-cell with $\alpha = \beta = \gamma = 1$, the average processing time on a Pentium 133Mhz machine for a CNN time-step is 30 seconds and a reasonable decrease in error during the learning process is achieved after 3 minutes. It follows that a 512x512 image will take 2 hours to be processed and about 12 hours will be necessary for learning. To speed-up the computation, a parallel hardware implementation of more than 1 cell is the best solution. In Fig.5. the schematic of the P-cell from the perspective of a fully parallel implementation is presented. Analog VLSI technology was considered since it offers the most convenient (in terms of occupied area) implementation of such operators as summation (by using the current Kirckhof's law), absolute value (using rectifying properties in PN junctions) and synaptic operators $\otimes_\alpha$ (e.g. by using CMOS switches). However, when accuracy is needed, the same schematic can be implemented via parallel digital hardware. Fig.6. presents a description of each building block (here also called "atoms").

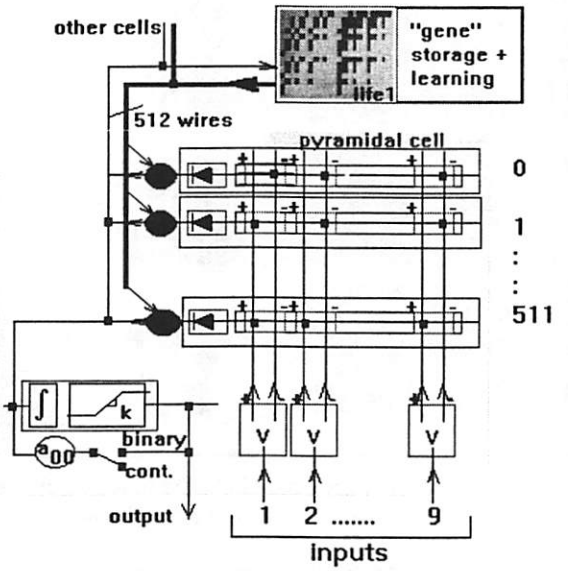

Fig.5: The implementation schematic of the P-cell.
The gene shown in this figure corresponds to the implementation of Conway's "Life" game.

The following features should be emphasised:

- Only one learning and parameter storage circuitry (the "gene" atom) is needed, regardless of the size of the image[5]. This offers important advantages for parallel implementation of multiple cells. Indeed, the more cells we have on the chip the less the percentage of area occupied by the learning and storage circuitry.
- In a fully parallel implementation, the fixed part of the truth table (the $a_{j,i}$ coefficients) are implemented as hard-wires and thus the occupied area is the smallest possible.
- Both rectifier and synaptic operators associated with each pyramidal atom can be implemented in a single VLSI cell with a small number of transistors. If the information processing at the cell output is restricted to the set {-1,0,1}, this atom may be a simple analog switch controlled by the resulting current contribution of the 9 inputs selected according to the defining coefficients $a_{j,i}$ of the pyramidal atom.
- Since the output of the cell is computed as a summation, it is convenient to consider the outputs of the "gene" atom (which represents the parameters g_j in the mathematical model) as currents, so that the summation can be realised in a single metal node, via KCL.
- In a real implementation, it is not necessary to implement nine grid-distributors for each cell. Instead, grid-distributors will be associated with the CNN inputs and, thus, each cell will need only one grid-distributor with 9 pairs of outputs. Thus, the number of transistors may be considerably reduced.

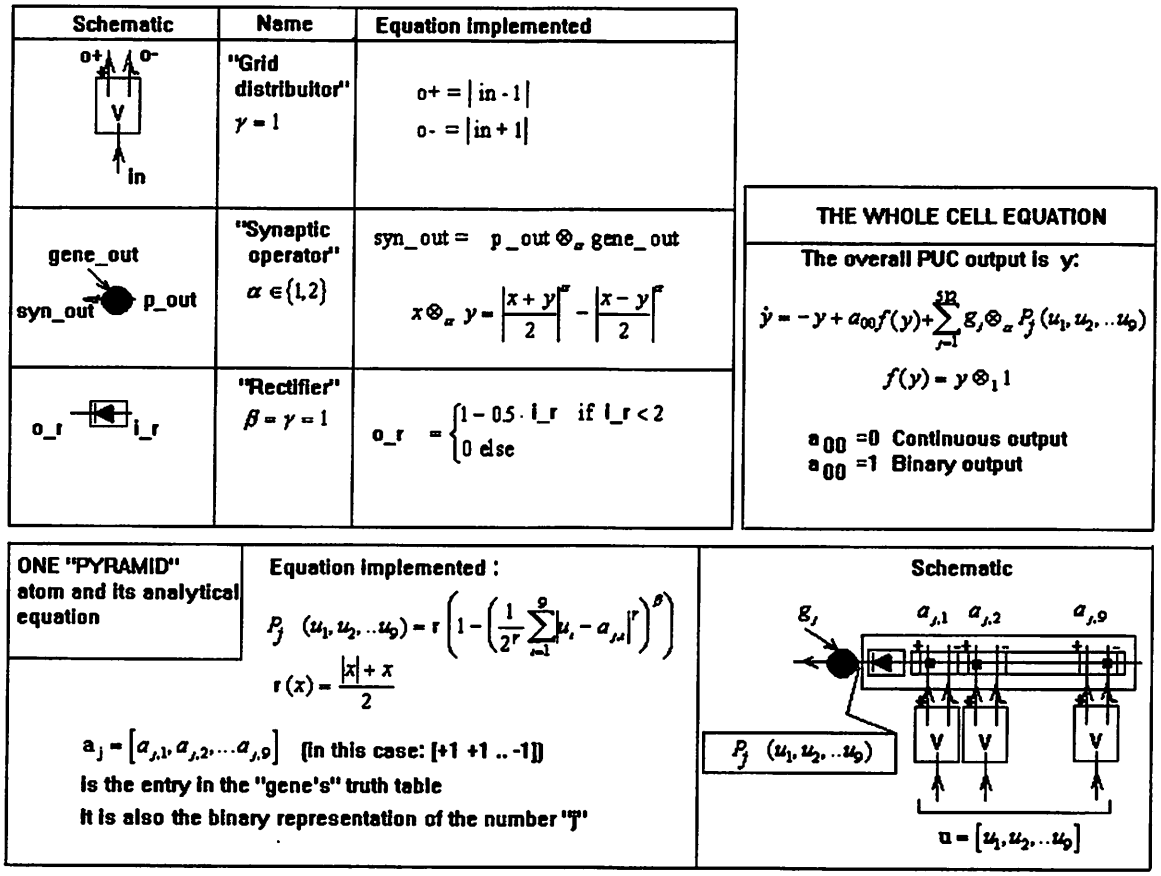

Fig.6: The building blocks (atoms) of the analog P-cell implementation

Although this implementation solution is fast (there are only three layers of physical devices between input and output) and needs a small number of basic building blocks, it has two disadvantages:

[5] Assuming a homogenous CNN.

1) A big current fan-out ($256 I_{ref}$) for each output of the current distributors, as a result of the fixed part section of the truth table. As long as small reference currents are associated with low speed, it follows that a compromise between computation speed and current consumption should be considered. For a reasonable consumption of 1mA / cell it follows that $I_{ref} \leq 1 / \left(\underbrace{512 \times 9}_{\substack{\text{the worst case} \\ \text{consumption in all} \\ \text{pyramidal atoms}}} \right) = 0.21 \mu A$.

2) A huge number of transistors, most of them needed as a result of 1). A rough estimate of this number is: $\underbrace{256 \times 18}_{\substack{\textit{distributors} \\ \textit{fan-out}}} + \underbrace{512 \times 4}_{\substack{\textit{rectifier+synaptic} \\ \textit{operators}}} + \underbrace{18+8}_{\substack{\textit{grid} \\ \textit{distributors}}} = 6682$ transistors / active cell.

It is interesting to observe however, that these disadvantages (which are in fact, the price paid for both universality and high speed) may be overcome using a sequential implementation, for example within the framework of a CNN Universal machine [16]. In this case, only one pyramidal atom per cell will be required. The hard-wired coefficients $a_{j,i}$ will be now replaced with analog (CMOS) switches digitally controlled by an external (the same for all cells) 9 bit counter. The output summation will be now made over time using an integrator of the output atom and assuming that both the counter and the integrator will be reset at the beginning of each 512-step computation cycle. The "gene" atom will now be a shift register controlled synchronously by the same clock as for the counter. Instead of 512 wires as in a fully parallel implementation it will have only one output connected to the synaptic operator of each cell. Thus, by sacrificing the computation time (512 times slower), it is possible to get a higher density. A rough estimate for the number of transistors gives in this case: $\underbrace{18 \times 4}_{\substack{\text{counter controlled} \\ \text{switches}}} + \underbrace{18+8}_{\substack{\text{grid} \\ \text{distributors}}} + \underbrace{4}_{\substack{\text{rectifier +} \\ \text{synaptic} \\ \text{operators}}} = 102$ transistors / cell.

This means an area reduction of about 65-times; and the current consumption per cell is also reduced 512 times. Since a 10 ns internal clock time is typical with actual technologies, it follows that such a cell will be able to process information at a speed of about $5 \mu s$. Assuming that 32x32 cells will be implemented on the same chip, it follows that there is still a considerably speed-up of $\underbrace{30 s}_{\textit{Pentium}} / \underbrace{5 \cdot 10^{-6} s}_{\substack{\text{VLSI} \\ \text{sequential}}} = 6{,}000{,}000$ times compared to a standard micro-processor implementation.

Conclusions

From the CNN perspective, using this Pyramidal Universal cell allows one to implement in a unified way all previously discovered linear and nonlinear templates which deal with binary imagery. Moreover, it offers a tremendous potential for investigating new processing functions and/or for learning new ones based on examples. A natural extension to grey level or colour image processing while preserving the logical meaning of the binary image processing is also possible.

References

[1] L.O. Chua, and S.M. Kang, "Section-wise piecewise-linear functions: canonical representation, properties, and applications", *Proceedings of IEEE,* Vol. **65**, June 1977, pp. 915-929.

[2] L.O.Chua, T. Roska, T. Kozek, Á. Zarándy, "CNN universal chips crank up the computing power", IEEE Circuits and Devices, July 1996, pp. 18-28.

[3] Cover, T.M., "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition", in *IEEE Trans. on Electronic Computers*, EC-14, pp. 326-334, 1965.

[4] K. Crounse, E.L. Fung, and L.O. Chua, "Efficient implementation of neighbourhood logic for cellular automata via the Cellular Neural Network Universal Machine", *IEEE Transactions on Circuits and Systems- I,* Vol. **44**, number 4, April 1997, pp. 355-361.

[5] R. Dogaru, K. Crounse, and L.O. Chua, "A generalized synaptic operator with application for efficient VLSI implementation of cellular neural networks", submitted to IEEE CAS-I Transactions, January 1997.

[6] R. Dogaru and L.O. Chua, "A VLSI-friendly synapse model and its application for fast and cheap signal classification problems with neuro-fuzzy networks", in Memorandum ERL M97/5, U.C. Berkeley, 16 January 1997.

[7] R. Dogaru, A.T. Murgan, S. Ortmann, M. Glesner, "A modified RBF neural network for efficient current-mode VLSI implementation", in *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems (Micro-Neuro'96),* IEEE Computer-Press, Laussane 12-14 Febr. 1996, pp. 265-270, 1996.

[8] M.H. Hassoun, "Fundamentals of Artificial Neural Networks", 1995, MIT Press

[9] Haykin, S., "Neural networks: A Comprehensive Foundation", 1994, New York, Macmillan.

[10] C. Kahlert, and L. O . Chua, "The complete canonical piecewise-linear representation-part I: the geometry of domain space", IEEE Trans. on Circuits and Systems-I, Vol. 39, No.3, March 1992, pp.222-236.

[11] T. Poggio, F. Girosi, "Networks for approximation and learning", *Proceedings of the IEEE,* Vol. **78**, September 1990, pp. 1481-1497.

[12] Widrow, B., and Hoff, M. E., Jr., "Adaptive switching circuits", in *IRE Western Electric Show and Convention Record*, part 4, pp. 96-104, 1960.

[13] L.A. Zadeh, "Fuzzy logic", *IEEE Computer,* Vol.1, pp. 83-93, 1988.

[14] L.O. Chua, C.A. Desoer, E.S. Kuh, "Linear and Nonlinear Circuits", 1987, McGraw-Hill.

[15] L.O. Chua, "Nonlinear and neural information processing", Lecture Notes EE-129, Spring-1997, EECS - U.C. Berkeley.

[16] T. Roska and L.O. Chua, "The CNN universal machine: an analogic array computer", *IEEE Trans. on Circuits and Systems -II,* Vol 40, pp. 163-173, March 1993

[17] T. Roska et al. "CSL-CNN software library (templates and algorithms) - V6.4.", Analogical and Neural Computing Laboratory, Hungarian Academy of Science, Budapest, 1995.