Proceedings of Sixth Italian Workshop on Parallel Architectures and Neural Networks. Vietri sul Mare, Italy, May 12-14, 1993,, World Scientific (E. Caianiello, ed.)

CELLULAR NEURAL NETWORKS: A REVIEW

VALERIO CIMAGALLI and MARCO BALSI
Department of Electronic Engineering, "La Sapienza" University of Rome
via Eudossiana, 18, I-00184, Rome, Italy
tel.: +36-6-44585836 fax: +39-6-4742647 E-mail: cima@tce.ing.uniroma1.it

ABSTRACT

A unified review of the Cellular Neural Network paradigm is attempted. First of all, general theoretical framework is stated, followed by description of particular models proposed in literature and comparison with other Neural Network and parallel computing paradigms. Theory of such systems, especially the issue of stability, is then resumed by listing main results available. Applications, design and learning follow. The paper is concluded by description of proposed and tested hardware realizations.

1. Cellular Neural Networks: spatially defined parallel analog computing for local and diffusion-solvable problems

Problems defined in space-time, e.g. image processing tasks, partial differential equations (PDE) systems, and so on, are often characterized by the fact that the information necessary to solve for the future or steady state of the system at a certain point is contained (from the start, or from a certain time on) within a finite distance of the same point. Therefore, these problems are solved by a relaxation and information diffusion process, which develops at the same time at all points of space domain.

Cellular Neural Network (CNN) is an analog parallel computing paradigm defined in space, and characterized by locality of connections between processing elements (cells, or neurons). Such systems are best suited for local and diffusion-solvable problems such as those considered above.

Two examples may help to give a first glance at CNN operation.

The first problem taken into consideration is halftoning of a grey-scale image. This kind of processing, used in newspaper photographs, Xerox and fax machines, is used to convert a continuously shaded image into one made of black dots on white background, that, when filtered by the eye (spatial low-pass) gives the same impression as the original image.

When a point of the image is considered, it is apparent that decision whether it should be black or white depends not only on the grey level of the original in the same point, but also on neighboring points grey level, and on decisions made for neighboring points. Therefore, parallel processing is best suited for this problem; however, sequential

filtering is generally applied on the scanned image, thereby introducing artefacts in the halftoned image. A CNN-halftoning system, developed by Crounse, Roska and Chua[1], might do the same job in parallel fashion at the speed of analog circuitry, with greater accuracy.

An example of halftoning is given in figure 1.

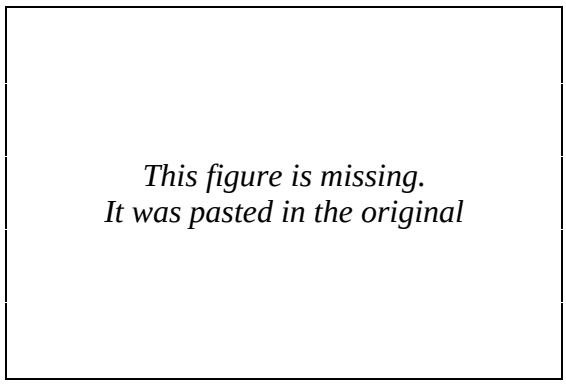


Figure 1 An example of halftoning

A second example problem, employing wider diffusion of information, is connected component detection (CCD), which can be used as a preprocessing for pattern recognition. It consists of counting the number of connected components found by scanning an image along a given direction. An example of CCD obtained by a CNN of an image along three different directions is shown in figure 2. This operation may be obtained by making the image drift towards a border, squashing it to 1-pixel width while preserving 1-pixel wide separation between disconnected parts. This can be done by only using local information at every instant of time. Evolution of CCD processing of a 1-dimensional image is depicted in figure 3.

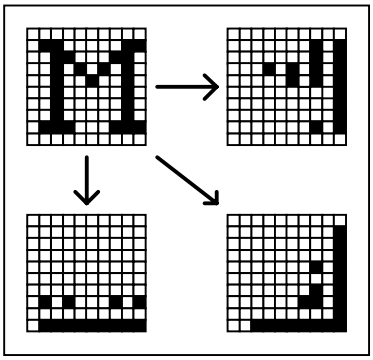


Figure 2 CCD result along three directions

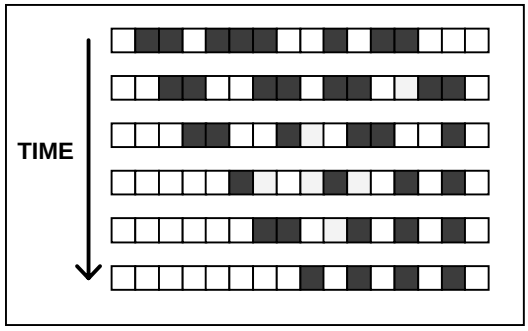


Figure 3 Time evolution of CCD on a 1-dimensional image. Grey squares represent non-saturated-state pixels.

The main difference between CNNs and other Neural Network (NN) paradigms is the fact that information is only exchanged between neighboring neurons. This characteristic does not prevent the capability of obtaining global processing, as the CCD example shows. By exploiting locality of connections, electronic IC and optical or electro-optical implementations become feasible, even for large nets, which is the main advantage of CNNs over NNs.

In the following section, a general definition of CNN is given, which is particularized to the different CNN models found in literature. Section 3 discusses stability of CNNs, and in section 4 a review of applications is given. The question of design and learning is then

confronted in section 5; hardware implementations follow in section 6. In the appendix tables of symbols and acronyms are written for convenient reference.

2. Defining Cellular Neural Networks

2.1 *Level 1 (system architecture)*

2.1.1 *Definitions*

A Cellular Neural Network is a system of cells (or neurons) defined on a normed space $\mathcal{C}$ (cell grid), which is a discrete subset of $\mathfrak{R}^n$ (generally $n\leq 3$), with distance function $d:\mathcal{C}\mapsto\mathfrak{I}$ ($\mathfrak{I}$ is the set of integer numbers). Cells are identified by indices defined in a set $\mathcal{I}$. One or several *neighborhood* functions N are defined as follows:

$$N_r:\ \mathcal{I}\rightarrow\mathcal{I}^\alpha$$
$$N_r(i)=\{j|d(i,j)\leq r\} \qquad (1)$$

where α depends on r (neighborhood size) and on space geometry (e.g. $\alpha=(r+1)^2$ on square 2-dimensional grid). Figure 4 shows examples of 2-dimensional CNNs and neighborhoods.

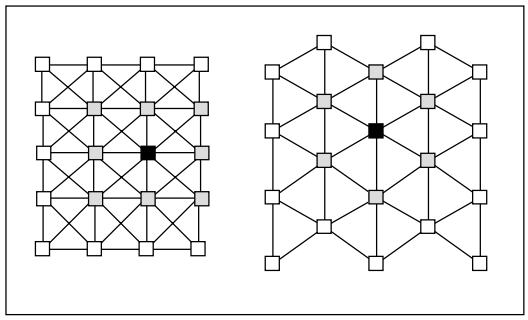


Figure 4 Examples of neighborhood of size r=1 in CNNs defined on a square or hexagonal grid $\mathcal{C}\subset\mathfrak{R}^2$. Grey cells belong to the neighborhood of black cells.

Cells are multiple input - single output nonlinear processors all described by one, or one among several different, parametric functionals. A cell is characterized by a state variable, that is generally not observable as such outside the cell itself. Every cell is only connected to cells within a neighborhood of itself.

2.1.2 *Comments*

When considered as a system, a CNN is a Neural Network characterized by the fact that connections are only allowed between neighboring neurons. The notion of distance implies that the network is intrinsically defined in space; generally only 1-, 2- or 3-dimensional space is considered, so that the CNN can be directly mapped into a physical realization scheme, that can profit of a dramatically simplified connection layout.

The cell grid can be e.g. a planar array (with rectangular, triangular, hexagonal geometry), a torus, a 3-dimensional array, generally considered and realized as a stack of 2-dimensional arrays (layers).

Cells may be all identical, or they can belong to a few different types (as is the case for biological neurons), and more than one connection network may be present, with different neighborhood size (short range interactions and subsystem connections). It is

obvious that, if the neighborhood size were as large as the network itself, we might obtain a fully connected network. It is understood that we shall not call such a net "cellular", and that generally the neighborhood shall have small size.

Cells may be very simple, or (moderately) complex. The "moderately" limit might be described by the fact that CNN dynamics must basically depend on information flow in the net, rather than on the operation inside the cells, so that we shall e.g. exclude that a parallel digital computer be a CNN.

2.2 *Level 2 (system operation)*

2.2.1 *Definitions*

The CNN is a dynamical system operating in continuous or discrete time. A general form of the cell dynamical equations may be stated as follows:

(continuous time)

$$\Delta_t x_j(t) = g\left[x_j(t)\right] + \sum_{k\in N_r(j)} \mathcal{A}_{\mu_j}\left(x_j|_{(t-\tau,t]}, y_k|_{(t-\tau,t]}; \boldsymbol{p}_j^A\right) + \\ + \sum_{k\in N_r(j)} \mathcal{B}_{\nu_j}\left(x_j|_{(t-\tau,t]}, u_k|_{(t-\tau,t]}; \boldsymbol{p}_j^B\right) + I_j(t)$$

$$y_j(t) = \mathcal{F}\left(x_j|_{(t-\tau,t]}\right) \qquad (2a)$$

(discrete time)

$$x_j(n+1) = g\left[x_j(n)\right] + \sum_{k\in N_r(j)} A_{\mu_j}\left(x_j|_{[n-m,n]}, y_k|_{[n-m,n]}; \boldsymbol{p}_j^A\right) + \\ + \sum_{k\in N_r(j)} B_{\nu_j}\left(x_j|_{[n-m,n]}, u_k|_{[n-m,n]}; \boldsymbol{p}_j^B\right) + I_j(n)$$

$$y_j(n) = f\left(x_j|_{[n-m,n]}\right) \qquad (2b)$$

In Eqs.2, x,y,u,I denote respectively cell state, output, input, and bias; j and k are cell indices; g is a local instantaneous feedback function, N is neighborhood function (if more than one neighborhood is defined, several similar sums are present), $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ are arrays of parameters, notation $z|_T$ denotes the restriction of function z(•) to interval T of its argument (i.e. the set of all its values). In Eqs.2a, t is time, Δ_t is a differential operator (e.g. d/dt), τ is memory duration time, $\mathcal{A}$ is (one out of several possible, identified by index μ_j) neighborhood feedback functional, and in the same way $\mathcal{B}$ is input functional, $\mathcal{F}$ is output functional; in Eqs.2b, A and B are the analogous functions for discrete time, f is output function, n is time and m memory duration.

Eqs.2 are written so as to be straightforwardly particularized. Therefore, they involve some general notations that are not strictly necessary. E.g., as feedback, input and output operators are functionals, it would be sufficient to substitute d/dt for Δ_t without loss of generality, but in this way it is more evident that higher order dynamics is also allowed.

2.2.2 *Comments*

Eqs.2 specify at an operational level the way of functioning of cells and connections (the latter are generally considered just as wires, lumping into the cell any dynamical behavior of theirs, e.g. transmission delays). Defined as such, cells can be characterized by a functional block diagram that is typical of neural network theory: figure 5 depicts a three-stage functional block scheme of a cell, composed of a generalized weighted sum (in general nonlinear, with memory), (leaky n-th order) integration, output nonlinear function/functional. It is apparent that this scheme is more general, so that what mostly distinguishes CNNs from other Neural Networks is the fact that all summations are performed within a neighborhood.

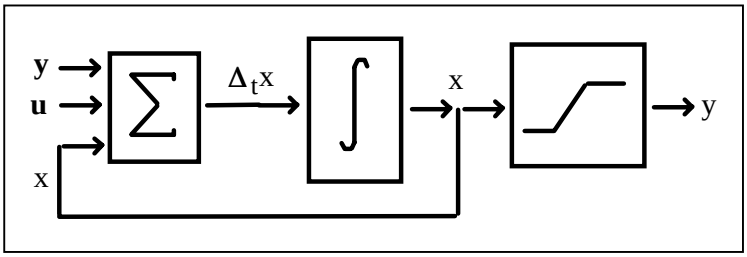


Figure 5 Functional block scheme of a CNN cell. Symbols are just reminders of more general operation.

Data can be fed to the CNN through two different ports: initial conditions of the state and proper input *u*. Bias values *I* may be used as a third port.

2.3 *Level 3 (special cases)*

Continuous time models (CT-CNN)

2.3.1 *Chua & Yang's CNN (CY-CNN)*

Chua & Yang's CNN[2] is a first order system, with linear instantaneous connections and piece-wise linear (PWL) output function (figure 6); Eqs. 2a specialize in this case as follows:

$$\frac{dx_j(t)}{dt} = -x_j(t) + \sum_{k \in N_r(j)} A_{jk} y_k(t) + \sum_{k \in N_r(j)} B_{jk} u_k(t) + I_j$$

$$y_j(t) = \frac{1}{2}\left(|x(t)+1| - |x(t)-1|\right) \qquad (3)$$

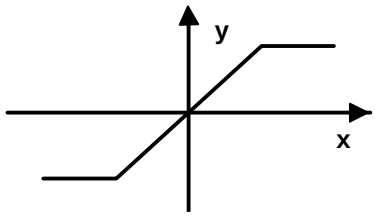


Figure 6 Piece-wise linear output function of CY-CNN

2.3.2 *Linear Cloning Template CNN (LCT-CNN)*

A particular case of Chua & Yang's CNN is obtained by making parameter values space-invariant[2]. Without loss of generality, we may consider a square 2-dimensional grid and use double indices to indicate position in the grid. In this case Eqs. 3 can be written for the cloning template CNN as:

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{kl\in N_r(ij)} A_{(i-k)(j-l)}(t)y_{kl} + \sum_{kl\in N_r(ij)} B_{(i-k)(j-l)}(t)u_{kl} + I$$

$$y_{ij}(t) = \frac{1}{2}\left(\left|x_{ij}(t)+1\right| - \left|x_{ij}(t)-1\right|\right) \qquad (4)$$

This means that A, B and I values can be determined by cloning template matrices that are identically repeated in the neighborhood of every neuron:

$$\mathbf{A} = \begin{bmatrix} A_{-1;-1} & A_{-1;0} & A_{-1;1} \\ A_{0;-1} & A_{0;0} & A_{0;1} \\ A_{1;-1} & A_{1;0} & A_{1;1} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} B_{-1;-1} & B_{-1;0} & B_{-1;1} \\ B_{0;-1} & B_{0;0} & B_{0;1} \\ B_{1;-1} & B_{1;0} & B_{1;1} \end{bmatrix} \quad I \qquad (5)$$

Such CNNs operate a kind of nonlinear convolution with the template.

LCT-CNN is the most widely studied CNN model (see e.g. bibliography of ref. 3), mainly for image processing applications. Uniformity of the network allows for easier design and realization.

2.3.3 *Full-Range CNN (FR-CNN)*

As will be shown in section 3 (theorem 1) CY-CNN state values are bounded by a limit that may become rather large, which is inconvenient for realization purposes. For this reason, Rodríguez-Vázquez *et al.*[4] proposed, with respect to CY-CNN, to move the nonlinear limiting from the output into state dynamics, therefore writing cell equation as follows:

$$\frac{dx_j(t)}{dt} = g\left[x_j(t)\right] + \sum_{k\in N_r(j)} A_{jk}x_k(t) + \sum_{k\in N_r(j)} B_{jk}u_k(t) + I_j$$

$$g(x) = \lim_{m\to\infty} \begin{cases} -m(x+1)+1 & x < -1 \\ -x & |x| < 1 \\ -m(x-1)-1 & x > 1 \end{cases} \quad (6)$$

In this way, state range is confined in the same range as input and output values, but behavior of the network is not qualitatively changed. The FR-CNN is very similar to a Brain-State-in-a-Box (BSB[5]) network, with only local connections.

In a similar way, a discrete-time FR-CNN may be defined.

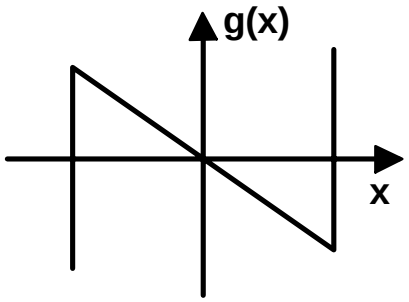


Figure 7 Local feedback function of the FR-CNN

2.3.4 *Nonlinear and Delay-Type CNN (NL-CNN/D-CNN)*

An extension of CY-CNN is obtained by allowing nonlinear and delayed connections, so that the following terms may be added to dynamical Eqs.3:

$$\sum_{k\in N_r(j)} \hat{A}_{ik}\left(y_j(t), y_k(t)\right) + \sum_{k\in N_r(j)} \hat{B}_{ik}\left(u_j(t), u_k(t)\right) \text{ (nonlinear connections)} \quad (7)$$

$$\sum_{k\in N_r(j)} A^{\tau}_{ik} y_k(t-\tau) + \sum_{k\in N_r(j)} B^{\tau}_{ik} u_k(t-\tau) \text{ (delayed linear connections)} \quad (8)$$

Nonlinear and delay-type terms may be made space-invariant, therefore defining nonlinear and delay-type cloning templates (NLCT/DCT-CNN[6]); more than one delayed term may be present at once, and delayed nonlinear connections may be also defined. NL- and D-CNNs have a wider scope of application than CY-CNNs, nonlinearity may allow e.g. for amplitude-selective connections; delay, besides modelling actual hardware delays, may be used for temporal processing such as motion detection.

2.3.5 *Linear Threshold/Feed-Forward CNN (LT/FF-CNN)*

Consider a 3-dimensional square grid CY-CNN written as follows, where we consider the lower two indices as indices within a (square grid) layer (which is a LCT-CNN), identified by the upper index, and where inputs are only applied to first layer:

$$\frac{dx_{ij}^{m}(t)}{dt} = -x_{ij}(t) + \sum_{kl \in N_r(ij)} A_{(i-k)(j-l)}^{mn}(t) y_{kl}^{n} + \sum_{kl \in N_r(ij)} B_{(i-k)(j-l)}(t) u_{kl} + I^{m}$$

$$y_{ij}^{m}(t) = \frac{1}{2}\left(\left|x_{ij}^{m}(t)+1\right| - \left|x_{ij}^{m}(t)-1\right|\right) \tag{9}$$

A layer is said to belong to the linear threshold class[7] if and only if

$$A_{ij}^{mm} = 0 \quad \forall (i,j) \neq (0,0) \tag{10}$$

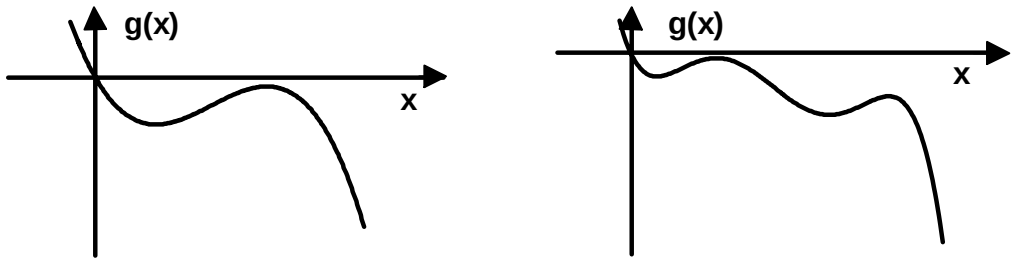
i.e. if no connections are present between cells of the same layer. The dynamics of cells of a linear threshold layer depend only upon their own states, the states of neighboring layers, and external inputs. Nossek, Seiler, Roska and Chua[8], add the notion of margin to the definition of LT-CNN, by imposing on local feedback weight condition $A_{jj}^{mm} = 1 + \mu_j$, where μ_j>0 is called margin.

A multilayer CNN is feed-forward[7] if and only if A^{mn}=0 for all n>m i.e., if intra-layer connections are only present from lower to higher index layer (no backward connections).

If all layers of a feed-forward CNN belong to the linear threshold class, and $A_{00}^{mm} = 1$ for all layers m>1, the steady state of each layer above the first depends only upon the input and its initial conditions. Such LT/FF-CNNs resemble a Multi-Layer Perceptron (MLP) where only local connections exist. In fact, as no feedback loop is present, cell states evolve monotonically to their steady value, obtained as a weighted sum of their inputs, just as a physical realization of a MLP would do when transmission delays and parasitic capacitance are taken into account.

2.3.6 *Polynomial CNN (P-CNN)*

A polynomial CNN[9] is defined by using an odd-degree polynomial as local feedback function *g*. General shapes of third and fifth degree polynomials are depicted in figure 8. Connections are nonlinear (step functions), chosen in a small set of possibilities in order to shape attractors for a pattern recognition device by synthesis. In fact, these functions allow directed flow of activation from cells corresponding to recognizable patterns towards cells corresponding to model (stored) patterns.



(a) **(b)**

Figure 8 Polynomial local feedback functions of degree 3 and 5 for the P-CNN

General equations for the P-CNN may be written as follows:

$$\frac{dx_j(t)}{dt} = g\left(x_j(t)\right) + \sum_{k \in N_r(j)} A_{\mu\, jk}\, y\left(x_k(t)\right) + u_j(t)$$
$$y_j(t) = x_j(t) \qquad (11)$$

Upon proper scaling of variables, dynamics of a P-CNN cell with third order local feedback is qualitatively analogous to that of CY-CNN cell (see ref. 2). Higher order polynomials raise the number of possible distinct equilibria of the cell.

2.3.7 *Non Uniform Processor/Multiple Neighborhood Size CNN (NUP/MNS-CNN)*

Motivated by neurophysiological evidence about the structure of brain cortex, a three-layer CNN memory architecture was introduced by Henseler and Braspenning[10]. This CNN has cells with different dynamics in the three layers, and neighborhoods have different sizes too, as shown in figure 9.

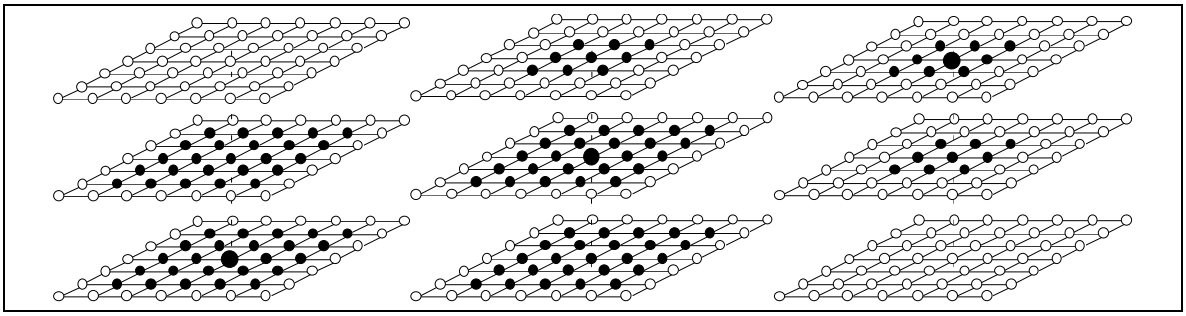


Figure 9 Henseler & Braspenning's NUP/MNS-CNN; black cells belong to the neighborhood of big black cells

Different cells allow for different processing in the three layers, while short- and long-range interactions make the CNN operate as a network of subsystems.

Other schemes of NUP/MNS CNNs were considered by Roska and Chua[6]. NUP-CNNs with two processor types may look as in figure 10, where black cells may be hidden processors (as proposed by Tan, Hao and Vandewalle[11], in order to increase storage capacity), and/or have different dynamics. Figure 11 shows an example of MNS-CNN, where fine and coarse grids are present.

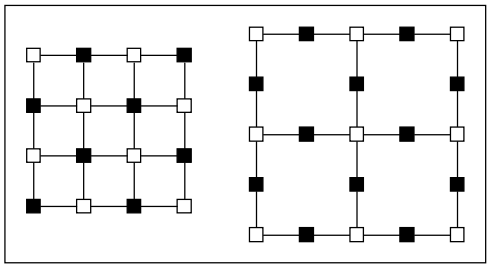


Figure 10 Examples of NUP-CNNs

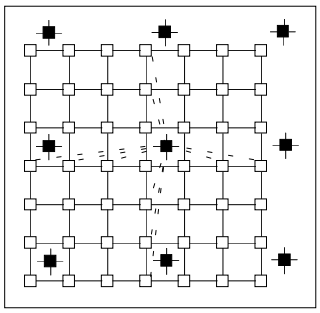


Figure 11 A MNS-CNN

2.3.8 *Moving Object Detecting architecture (MODA)*

MODA[12] (Cimagalli, Bobbi and Balsi, 1993) is a NUP-CNN with complex cells and periodic multiple cloning template structure. In fact, it is defined on a square grid with a 3×3 repeating pattern, as shown in figure 12. Black cells are called "central", white cells are "off-center". Off-center cells have two inputs, coming form two neighboring pixels of an image; their equations write as follows:

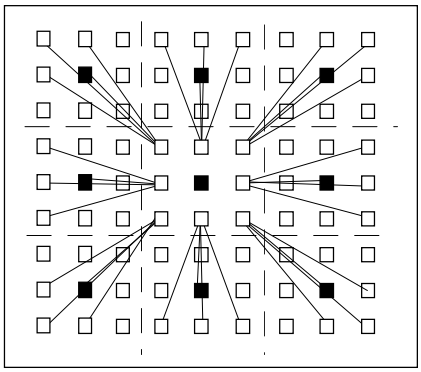


Figure 12 Cells are represented by white and black boxes. Connections towards a 9-neuron cluster are shown.

$$\frac{dx_j(t)}{dt} = -x_j(t) + \sum_{k \in N_{\mu_j}(j)} A_{\nu_j} y_k + B\left(u_{1j}(t-\tau), u_{2j}(t)\right) + I$$

$$y_j(t) = \gamma\sigma[x(t)] \tag{12}$$

where:

$$\sigma(x) = 0 \text{ if } x < 0 \text{ else } \sigma(x) = 1 \quad \text{(unit step)}$$

$$B(x,y) = \sigma\{\sigma(x-\varepsilon) + \sigma[(\alpha-1)x + y] + \sigma[(\alpha+1)x + y] + \sigma(y-\varepsilon) - \beta\} \tag{13}$$

α,β,γ and ε are suitable parameters. Feedback coefficients A depend on cell location, as can be deduced from figure 12.

Central cells have no dynamics: they just process their single input instantaneously:

$$y_j = B\left(u_j(t-\tau), u_j(t)\right) \tag{14}$$

With proper choice of parameters, MODA operates as a moving object detector. Cell state is associated to an element of trajectory (between neighboring pixels); input function detects permanence of a similar input in successive positions and times (spatio-temporal coincidence), while first order dynamics acts as a short term memory.

2.3.9 *Membrain, Relaxation Oscillator CNN (RO-CNN)*

Membrain[13] is a CNN with second order dynamics defined on a 2-torus. This means that borders of the net are connected to their opposites, as shown in figure 13.

For this network, Eqs. (2a) are particularized as follows:

$$\frac{d^2 x_j(t)}{dt^2} + \alpha \frac{dx_j(t)}{dt} = \sum_{k \in N_r(j)} A_{jk} y_k(t) + I_j(t)$$
$$y_j(t) = x_j(t) \tag{15}$$

Due to its toroidal structure, Membrain can perform translation-invariant pattern recognition. It solves a discrete-space generalized wave equation, where inputs, given only as initial states, act as a perturbation setting up oscillation of the system, which behaves similar to an elastic membrane. It is obvious that the same dynamics can be defined on a plane grid; in this case the usual reflection phenomena of wave theory appear at the borders.

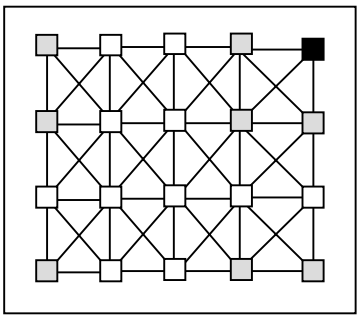


Figure 13 Neighborhood of a corner pixel in a CNN defined on a 2-torus

Another model of oscillator network[14] employs relaxation oscillators:

$$\frac{dx_j(t)}{dt} = \sum_{k \in N_r(j)} A_{jk} y_k(t) + u_j(t)$$
$$x_j(t) = f\left[y_j(t)\right] \tag{16}$$

Output *y* is defined implicitly through function *f*, represented in figure 14, where *y* can be taken to represent current flowing through a piece-wise linear current-controlled negative resistor.

This structure exhibited multiple attractor periodic and chaotic behavior, that might be exploited for Content-Addressable Memory (CAM) realization.

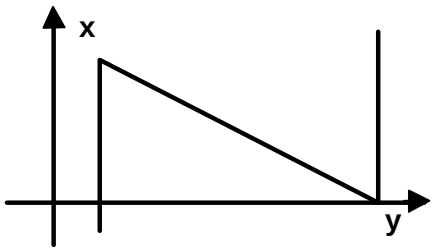


Figure 14 Implicit output function of the RO-CNN

2.3.10 *CNN with Local Logic (CNNL) - Analog Software Universal Machine*

CNNs may be expanded[3,15] by adding to every cell a few logical components, thereby making it a parallel "analogic" (analog and logic) universal computing machine, capable of executing analog software, i.e. applying several cloning templates in succession, in order to achieve complex tasks. This kind of processing (dual computing), may be regarded as the analog counterpart of Single-Instruction-Multiple-Data parallel computers. To do this, a local (analog and) digital memory is added, and a simple control logic, itself controlled by a global control unit.

A CNN of this type (analog software universal machine[3]) can be considered as a time-varying cloning template architecture, and its dynamical equations written as follows for linear templates:

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{kl\in N_r(ij)} A_{(i-k)(j-l)}(t)y_{kl}(t) + \sum_{kl\in N_r(ij)} B_{(i-k)(j-l)}(t)u_{kl}(t) + I(t)$$

$$y_{ij}(t) = \frac{1}{2}\left(\left|x_{ij}(t)+1\right| - \left|x_{ij}(t)-1\right|\right) \qquad (17)$$

Local memory and logic may also be used to apply simple logical processing to binary valued outputs, before feeding them back to CNN input[16].

2.3.11 *Relation with other continuous-time Neural Network models and systems*

Among NN paradigms, the continuous Hopfield model (CH-NN[17]) most closely resembles CNNs. In fact, it has substantially the same dynamics of CY-CNN, in the limit of maximum size neighborhood:

$$\frac{dx_j(t)}{dt} = -\alpha x_j + \sum_k A_{jk}y_k + I_j$$

$$y_j = f\left(x_j\right) \qquad (18)$$

where f is a sigmoid function, e.g.

$$f(x) = \frac{1}{1 - e^{-\beta x}} \tag{19}$$

The fact that *f* is not piece-wise linear makes no significant difference in the overall behavior of the network. Actually, a PWL function can be approximated to any precision by a continuous function, and sometimes this substitution is done in CNNs because, besides being more realistic from the point of view of realization, is also useful in theoretical proofs.

The CH-NN is used as a nearest-neighbor encoding CAM, with the cue given as initial state; therefore, no input is present.

Theory concerning CH-NNs can obviously be applied to CNNs too, but, as the restriction of locality of connections endows CNN dynamics with peculiar properties, this similarity is only seldom used, and most results of CNN theory have been obtained independently.

It is also interesting to consider the similarity between CNNs and (physical) systems described by systems of partial differential equations (PDE). In fact, these two kinds of systems share the property that dynamic behavior only depends on local spatial interactions[2].

In fact, consider divergence (∇) and Laplace (∇^2) vector differential operators, which are basic building blocks of fundamental equations of physics (e.g. heat, Poisson, Navier-Stokes equations ...); in two dimensional rectangular coordinates they are written as:

$$\begin{aligned} \nabla &= \frac{\partial}{\partial x} + \frac{\partial}{\partial y} \\ \nabla^2 &= \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \end{aligned} \tag{20}$$

If we approximate space derivatives numerically, e.g. by using incremental ratios, it is apparent that by taking a regular discretization of space with steps h_x and h_y, the following cloning template A matrices for LCT-CNN implement the same operators, as long as all cells operate in the linear region of output function (obviously different numerical derivation formulas may yield different neighborhood size and coefficients):

$$\mathbf{A}_{\nabla} = \begin{bmatrix} 0 & \frac{1}{2h_y} & 0 \\ \frac{-1}{2h_x} & 1 & \frac{1}{2h_x} \\ 0 & \frac{-1}{2h_y} & 0 \end{bmatrix} \quad \mathbf{A}_{\nabla^2} = \begin{bmatrix} 0 & \frac{1}{h_y^2} & 0 \\ \frac{1}{h_x^2} & \left(1 - \frac{2}{h_x^2} - \frac{2}{h_y^2}\right) & \frac{1}{h_x^2} \\ 0 & \frac{1}{h_y^2} & 0 \end{bmatrix} \tag{21}$$

As a first-order system, CY-CNNs approximate a generalized heat equation; in the same way, Membrain, a second-order system, implements a generalized wave equation.

Discrete time models

2.3.12 *Discrete Time CNN (DT-CNN)*

The discrete time version[11,18] of CY-CNNs is described by the following equations:

$$x_j(n+1) = \sum_{k \in N_r(j)} A_{jk} y_k(n) + \sum_{k \in N_r(j)} B_{jk} u_k(n) + I_j$$
$$y_j(n) = f\left[x_j(n)\right] \tag{22}$$

where f is the $sign(\bullet)$ function:

$$f(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases} \tag{23}$$

or the usual PWL function as in Eqs. 3[19] .

DT-CNNs may be used with linear cloning templates[18], as well as with nonlinear and time-varying templates[19].

2.3.13 *Multi-Valued CNN (MV-CNN)*

The multi-valued neuron for discrete time CNN was introduced by Aizemberg and Aizemberg[20]. The state and output of such cells is defined in complex space, therefore dynamical equations for the MV-CNN do not readily fit in the framework of Eqs. 2b. However, they can be written in similar form:

$$x_j(n+1) = \sum_{k \in N_r(j)} A_k y_k + I_j$$
$$y_j = csign(x_j) \tag{24}$$

Function $csign(\bullet)$ is defined as follows ($\boldsymbol{i}$ is the imaginary unit):

$$csign(z) = e^{\mathbf{i} \cdot 2\pi \cdot j/k} \quad \text{if} \quad 2\pi \cdot j/k \leq \arg(z) < 2\pi \cdot (j+1)/k$$
$$csign(0) = 1 \tag{25}$$

2.3.14 *Relation with other discrete-time neural network models and systems*

As discussed for CT-CNNs, also DT-CNNs may be considered as restrictions of fully connected NNs: Additive Grossberg NNs[21] have positive weights, negative local feedback and step nonlinearity, Discrete Autocorrelators[22] also have step nonlinearity, BSB has PWL nonlinearity, as for the FR-CNN. For these networks, however, the same comment applies about the fact that theory has just limited interest in CNN context.

It is instead more interesting to compare LCT-(DT-)CNN(L) with two other parallel computing paradigms, namely Systolic Arrays (SA) and Cellular Automata (CA). Table 1[23] compares the data type and processing specification for these models:

Table 1 Confronting CA, SA, and CNN

	Cellular Automaton	Systolic Array	CNN
data	logic values (1-4 bits)	numerical values (8-32 bits)	analog values
specification	truth table	numerical algorithm	cloning templates (analog software)

Analogies and differences between CA and CT-CNNs have been discussed and exploited by Chua and Shi[7].

Operation of the CA consists of two phases: a processing phase and a (local or global) propagation phase. These two phases (with local propagation) are also present in DT-CNN realizations[18], and can also be replicated in CT-CNNs, even if the latter operate asynchronously. In fact, Chua and Shi[7] showed that a LT-CNN can implement one iteration of a CA involving only local propagation, and that a LT/FF-CNN can also implement global propagation operation.

Therefore, it is possible to exploit CA design rules in the design of CNNs, in order to obtain the same functionality, with the advantage of easier realizability of the analog CNN chip.

In CA theory, the game of life algorithm has particular importance, because of its universality; Chua, Roska, Venetianer and Zaràndy[19] have discovered LCT-CNN templates for one step of the algorithm and a DT-CNN(L) analog algorithm to perform the continuing game, thereby proving universality of CNNLs.

3. Stability

Main theoretical results concerning CNN dynamics are hereafter reviewed. For proofs of theorems, please refer to quoted papers.

Theorem 1 (boundedness of state for CY/NL/D-CNNs)

All states x of a CY/NL/D-CNN are bounded for all time by

$$x_{\max} = 1 + \max_{j \in \mathcal{CY}} \left[\sum_{k \in N_r(j)} \left(\left|A_{jk}\right| + \left|B_{jk}\right| + \max_{y_1, y_2 \in [-1,1]} \left|\hat{A}_{jk}(y_1, y_2)\right| + \right.\right.$$

$$\left.\left. + \max_{y_1, y_2 \in [-1,1]} \left|\hat{B}_{jk}(y_1, y_2)\right| + \left|A^{\tau}_{jk}\right| + \left|B^{\tau}_{jk}\right| + \left|I_j\right| \right) \right] \qquad (26)$$

provided that nonlinear connection functions and initial conditions are bounded by 1. Moreover, ω-limit points of x ($x_j(\infty)$) are bounded by $x_{\max}$-1.[2,6,24,25]

Theorem 2 (stability of CY-CNNs with symmetric connections)

A CY-CNN with symmetric connections (i.e. $A_{jk} = A_{kj} \ \forall j,k \in \mathcal{CY} | k \in N_r(j)$) is asymptotically stable[2].

Theorem 3 (stability of positive cell-linking CY/NL-CNNs)

A NL-CNN satisfying the following condition (positive cell-linking - PCL - condition), is asymptotically stable, except possibly for a set of initial conditions of zero measure.

$$\text{a)} \ \forall j,k \in \mathcal{CY} | k \in N_r(j): \quad \frac{\partial \hat{A}_{jk}(y_j, y_k)}{\partial y_k} > 0 \text{ or } \hat{A}_{jk} \equiv 0;$$

$$\text{b)} \ \forall j,k \in \mathcal{CY} \ \text{ there exists a path in } \mathcal{CY} \ \{i_0 = j, i_1, \ldots, i_n = k\} \left| \hat{A}_{i_m i_{m+1}} \neq 0 \right. \qquad (27)$$

This is true in particular of PCL CY/LCT/NLCT-CNNs. For CY/LCT-CNNs, condition a) means that weights must be non-negative[6,26].

Theorem 4 (stability of opposite-sign LCT-CNNs)

A one-dimensional LCT-CNN with cloning template matrix $\boldsymbol{A} = (a_{-1}\ a_0\ a_{+1})$, with $a_{-1}a_{+1}<0$, $a_0>1$ is asymptotically stable if and only if $|a_{-1}|<a_0$-1 or $|a_{+1}|<a_0$-1 [27,28].

Theorem 5 (stability of positive strictly sign-symmetric CY-CNNs)

A CNN is positive if $A_{jk} \geq 0 \quad \forall j,k \in \mathcal{CY} | j \neq k$; it is strictly sign-symmetric if whenever $A_{jk} \neq 0$: $A_{jk}A_{kj} > 0$. A positive strictly sign-symmetric CNN is stable almost everywhere[29].

Theorem 6 (stability of acyclic CY-CNNs)

A CNN is acyclic if it has no cycles, i.e. if there exists no set of indices $\{i_1,i_2,...,i_n\}$, all distinct except i_n=i_1, such that $\forall j = 1,2,\ldots,n-1$: $A_{i_j i_{j+1}} \neq 0$.

An acyclic CNN with $A_{jj} > 1 \; \forall j \in \mathcal{C}$ is asymptotically stable[29].

Remark 1 (state space transformations extending stability results for CY-CNNs)

Consider a state space transformation of the form $x'=\mathbf{J}x$, with

$\mathbf{J} = diag\left((-1)^{n_1}, (-1)^{n_2}, \ldots, (-1)^{n_k}\right)$ and $n_i \in \{0,1\}$. Let $\tilde{\mathbf{A}} = \left(\left(A_{jk}\right)\right)$. If the transformed CNN, with state x', and weights obtained from $\tilde{\mathbf{A}}' = \mathbf{J}^{-1}\tilde{\mathbf{A}}\mathbf{J}$, is stable, so is the original CNN. Chua and Wu[29] specify possible different transformations for LCT-CNNs.

Theorem 7 (binary output property of CY-CNNs)

In a CY-CNN, if $A_{jj} > 1 \; \forall j \in \mathcal{C}$, then any stable equilibrium state must have all components with magnitude greater than 1, i.e. $\left|x_j(\infty)\right| > 1 \; \forall j \in \mathcal{C}$[2].

Theorem 8 (global asymptotic stability of low-feedback CY-CNNs)

A CY-CNN is globally asymptotically stable (i.e. it has a single asymptotically stable equilibrium point) if it is asymptotically stable and $A_{jj} < 2 \; \forall j \in \mathcal{C}$[30].

Theorem 9 (global asymptotic stability of low-self-feedback CY-CNNs)

A CY-CNN is globally asymptotically stable if

$$A_{jj} + \frac{1}{2} \sum_{k \in N_r(j)} \left(\left|A_{jk}\right| + \left|A_{kj}\right|\right) < 1 \; \forall j \in \mathcal{C}$$[31].

Theorem 10 (stability of D-CNNs)

In a D-CNN, let $\tilde{\mathbf{A}} = \left(\left(A_{jk}\right)\right)$, and $\tilde{\mathbf{A}}^{\tau} = \left(\left(A_{jk}^{\tau}\right)\right)$. If: $\tilde{\mathbf{A}}$ is non-negative in the off-diagonal locations; $\tilde{\mathbf{A}}^{\tau}$ has only non-negative elements; $\tilde{\mathbf{A}} + \tilde{\mathbf{A}}^{\tau}$ is an irreducible matrix; and the set of equilibrium points is finite, then the network is almost everywhere asymptotically stable[24].

Theorem 11 (stability of D-CNNs)

Let $\boldsymbol{P} = -\boldsymbol{1} + \tilde{\mathbf{A}} + \tilde{\mathbf{A}}^{\tau}$ ($\boldsymbol{1}$ is the identity matrix). If $\tilde{\mathbf{A}}^{\tau}$ is invertible, $\boldsymbol{P}$ is symmetric, and $\left\|A^{\tau}\right\| < \frac{2}{3\tau}$ (where $\|\bullet\|$ denotes the Euclidean norm), the corresponding D-CNN is asymptotically stable[32].

Theorem 12 (global asymptotic stability of D-CNNs)

If a D-CNN is such that: $\tilde{\mathbf{A}}$ has only non-negative off-diagonal elements, $\tilde{\mathbf{A}}^{\tau}$ has non-negative elements, and $-\left(\tilde{\mathbf{A}}+\tilde{\mathbf{A}}^{\tau}\right)$ is row-sum dominant, i.e.

$\sum_{k\in N_r(j)} -\left(A_{jk}+A_{jk}^{\tau}\right) \geq 0 \ \forall j \in \mathcal{CB}$, it is globally asymptotically stable[25].

Theorem 13 (global asymptotic stability of NL/D-CNNs)

In a NL/D-CNN, let $\tilde{\mathbf{A}}_{nl}(f_1(\mathbf{y})) = \left(\left(\hat{A}_{jk}\left(y_j, y_k\right)\right)\right)$. If $\tilde{\mathbf{A}}_{nl}$ satisfies a Lipschitz condition with constant L: $\left|\tilde{\mathbf{A}}_{nl}(\mathbf{y}_1)-\tilde{\mathbf{A}}_{nl}(\mathbf{y}_2)\right| \leq L|\mathbf{y}_1-\mathbf{y}_2| \ \forall \mathbf{y}_1, \mathbf{y}_2$, and $L+\left\|\tilde{\mathbf{A}}^{\tau}\right\|<1$, then the NL/D-CNN is globally asymptotically stable[25].

Theorem 14 (stability of eigendominant DT-CNNs)

In a DT-CNN, let $b_j = \min_u \left|\sum_{k\in N_r(j)} B_{jk}u_k + I_j\right|$. If $\forall j: A_{jj} \geq 0$, and $A_{jj}+b_j > \sum_{k\in N_r(j)} \left|A_{jk}\right|$, the network converges in a maximum number of time steps which is equal to the number of cells in the network[18].

Theorem 15 (stability of symmetric DT-CNNs)

A DT-CNN with symmetric feedback coefficients $A_{jk} = A_{kj}$ either converges, or oscillates with a period of two[18].

4. Applications

Since their introduction, a wide range of applications has been proposed and tested for CNNs. In this section, only a synthetic review of these applications can be given, for reason of space; detailed description can obviously be found in quoted papers.

4.1 *Applications of LCT-CNNs*

LCT-CNNs have found most applications in image processing, also profiting of filter theory (e.g. ref. 33, 34). Besides local processing, global effect are obtained by information propagation, also by properly setting both input and initial state (e.g. CCD, hole-filler). CNNs may be used as preprocessors for pattern recognition tasks, also in connection with other kinds of NNs and with logical computers (e.g. feature detection, character recognition). Table 2 summarizes the most significant contributions.

Table 2 LCT-CNN applications

Application	Reference
line, edge and corner extraction	33
noise removal (low-pass filtering)	33
connected component detection	35
hole filling	35
shadow detection	35
image thinning/thicking	35, 36
small object isolating (for counting)	37
halftoning and information compression	1,38
character recognition	35, 39, 40

Table 2 (continued)

printed circuit layout error detection	41
vehicle guiding	42
Radon transforming	7

4.2 *Applications of NLCT-CNNs*

NLCT-CNNs have been applied to image processing (grey-scale image contour detection[43]); simulation of physical and biological systems (lattice equations, CNN-retina). Table 3 lists more contributions.

Table 3 Applications of NLCT-CNNs

Application	Reference
Grey-scale image contour extraction	44
image thicking	44
convex corner detection	45
simulation of lattice equations	46
CNN-retina	47
local Boolean function evaluation	48

4.3 *Applications of other CT-CNNs*

Delays have been exploited in order to detect moving parts of images, by confronting subsequent snapshots (D-CNNs, MODA); analog software for CNNLs has been developed for the solution of complex tasks, among them the game of life, which proves universality of CNNL machines. Globally asymptotically stable CY-CNNs may be applied to approximation of mappings between continuous or discrete vector spaces. Other applications are listed in table 4.

Table 4 Applications of other CT-CNNs

Application	Reference
motion detection	12, 49
mapping approximation	31, 50
game of life	19
artificial vision (stereopsis)	51
object counting	19
translation-invariant pattern recognition	13
oscillators, chaos generators	52

4.4 *Applications of discrete time CNNs*

Several authors have discussed use of discrete time CNNs as CAM; concerning image processing, DT-CNNs cloning templates have been found for many tasks, including some new functionalities in which speed of processing can be usefully controlled. A list of these applications is given in table 5.

Table 5 Applications of discrete time CNNs

Application	Reference
content-addressable memory (CAM)	11, 53, 54, 55
CCD, concentric contouring, minimal distance testing, image thinning and thicking	18
shadow detector	56

5. Design and learning

As stated in section 1, CNN dynamics is substantially governed by local interactions between cells, that just perform a quite simple processing. Therefore, as is usual in NN theory, CNN design involves as a basic step the choice of (generally a lot of) parameters (connection weights). It is to be noticed that cloning template CNNs have a very peculiar characteristic among NNs; in fact, they can be described by a small number of weights (e.g. a square $\mathcal{C}\mathcal{U}$ LCT-CNN with r=1 has 19 independent weights).

The choice of suitable parameters is configured as an optimization task, that can be accomplished in two extreme ways[53]: (1) given a complete and precise formulation of the problem to be solved, *synthesis* is performed by choosing systematically a nominal set of optimal parameters; (2) given a vague description of the task through a set of examples, *learning* is accomplished by presenting examples to the network, adjusting parameters with a suitable algorithm, that converges to an (at least local) optimum set. Proper learning involves generalization of functionality from a limited training set; however, we shall consider as learning also pattern storage (i.e. all examples given) and *design by examples*, where the task is also defined in terms of examples, but design is done by a systematic (non-adaptive) direct choice of parameters.

The first steps of CNN theory were done by obtaining useful LCT-CNN cloning templates by a sort of empirical synthesis, based on experience in other fields (e.g. filtering) and on trial-and-error. Most design theory for CNNs has been developed quite recently, and has concentrated in particular to cloning template CNNs. A brief review of the various methods proposed is given below.

Analog software, recently rigorously defined by Roska and Chua[15], opens the way to analog ("analogic") programming, which can be as an extension of CNN synthesis beyond the hardware level.

5.1 *Design by synthesis*

Synthesis methods for LCT-CNNs have been proposed by Chua and Thiran[57] and Osuna and Moschytz[58]. Both algorithms involve building a system of inequalities describing solution space, and solving them by a standard method (e.g. relaxation). The first paper gives sufficient conditions on template parameters that guarantee correct functioning; applying these conditions, however, is only feasible for rather simple problems and small number of free parameters. The second method quoted, aimed at finding symmetric templates satisfying the binary output condition (theorem 7) consists of a set of inequalities to be imposed exhaustively on desired and forbidden input/final output couples, which can be considered as restricted to neighborhood size. Slot[59] distinguishes two aspects of the feature detection problem: recognition of presence of the feature in input pattern, and propagation of information about this recognition. Cloning template matrices $\boldsymbol{B}$ and $\boldsymbol{A}$ are then designed so as to satisfy two sets of inequalities describing the two mentioned tasks. Chua and Shi[7] exploited analogy with CAs to design a LCT-CNN implementing the Radon transform; Crounse, Roska and Chua[38] designed halftoning templates by adapting parameters of known filters.

Synthesis of CY-CNN weights was considered by Seiler, Schuler and Nossek[54]. They gave a rigorous mathematical framework to the problem of obtaining robust solution to pattern storage problems, by stating sufficient conditions on nominal and actual values of parameters, written in terms of inequalities to be solved by a standard method, e.g. the simplex algorithm. They also give practical guidelines to apply the theory in the design of a CAM with given stable equilibria.

Realization of locally-defined Boolean functions by time-varying cloning template DT-CNNs is solved by Galias[49]: simple rules are given to set the weights from a product-of-sums or sum-of-products definition of the function.

For P-CNNs, used as CAM, an explicit synthesis procedure is given to choose nonlinear connection functions to store given patterns[9]. MODA, a dedicated architecture is also designed by explicit synthesis, implementing constraints of the problem[12].

5.2 *Learning*

LCT-CNNs have much less parameters than general CNNs and NNs. Therefore, learning for those networks has been accomplished with several optimization algorithms that are generally impractical for other models.

Zou, Schwarz and Nossek[60] obtain straightforwardly from examples a system of linear inequalities, to be solved by a relaxation method; the cost function is identified as the sum of distances of the current solution from separation planes defined by these inequalities. The same system is solved by Szolgay and Kozek[61] by the simplex algorithm.

Schuler, Nachbar, Nossek and Chua[62] taught a LCT-CNN state space trajectories by using backpropagation-through-time algorithm with conjugate gradient descent. Cost function is taken as an integral of distance between actual and desired solution over the entire trajectory of state evolution.

A different approach is used by Kozek, Roska and Chua[63], who employed a genetic algorithm, which can be easily applied to complex tasks too, due to the simple way of giving constraints and optimization criteria to be enforced.

CY-CNNs can be trained by recurrent back-propagation, which is a generalization of the well-known back-propagation algorithm to non-feed-forward networks. Operation of this algorithm is time-consuming when simulated, but it might be directly realized in hardware[30,31,64]. Hansen[65] employed the Boltzmann machine algorithm, maximizing *a posteriori* probability of correct output by gradient decent.

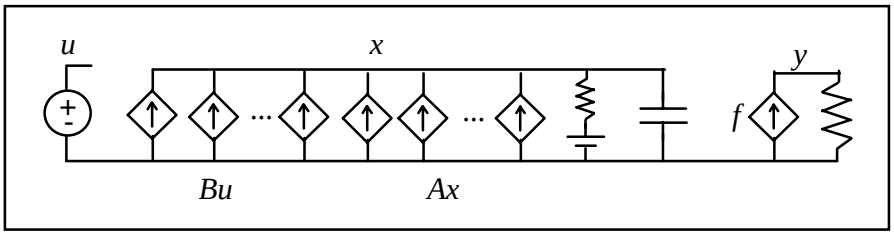
Concerning CNN-CAMs, Tan, Hao and Vandewalle[11] applied Hebb's rule and computed capacity of the network. Mizutani[56] also used Hebb-like storage. Aizemberg and Aizemberg[20] adapt weights for the MV-CNN in an a iterative manner, resembling a gradient descent, which finds a solution in a finite number of steps.

Learning for DT-CNNs was considered by Harrer, Nossek and Zou[57], who employed a relaxation method, as described for CT-CNNs, and by Magnussen and Nossek[66]. In the latter paper, a new algorithm called LASTNERD is defined, that employs line searches in order to minimize a cost function obtained as mean square distance from training examples. Training examples, and direction of line searches are chosen at random, so as to introduce a noise component into the process, that enhances robustness and reliability of the solution.

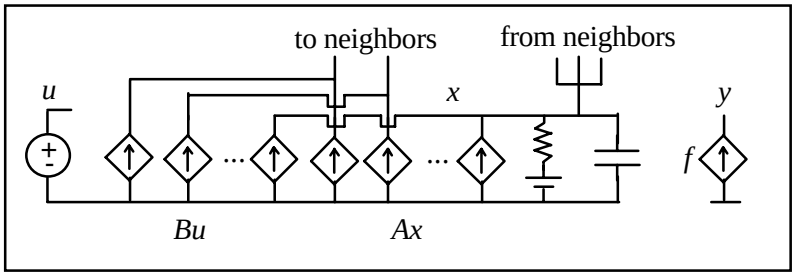
6. Hardware implementation

CY-CNN equations (3) are readily translated in a circuit scheme by interpreting cell state as voltage across an RC dipole realizing first-order dynamics. Connections are realized by voltage-controlled current sources (VCCS), so that summations are performed by injecting currents into the same node.

Based on this scheme (figure 15a), several implementations were proposed employing operational transconductance amplifiers (OTA) as basic blocks. Weighting of signals is performed at the output instead of input of cells, together with nonlinear output function. Therefore, the cell has multiple outputs and a single input instead of single output and multiple inputs (figure 15b).



(a)



(b)

Figure 15 Alternative models of CNN cell: (a) Chua & Yang's model, with bias represented as voltage instead of current; (b) dual model with weighting at the output

To do this, OTAs are used both in their linear operation range and in saturation, exploiting their transfer function, that approximates a PWL sigmoid very well. In this way, Cruz and Chua[67] designed a cell composed of three OTAs and two capacitors (figure 16). In this circuit, Cx is state capacitor, Cu is loaded with cell input, OTA A implements all current sources controlled by state voltage, and PWL output function. OTAs B and R work in their linear range; the first implements all current sources controlled by input voltage, the latter works as state resistor (with voltage bias). A 6×6 prototype was realized in 2μm CMOS technology.

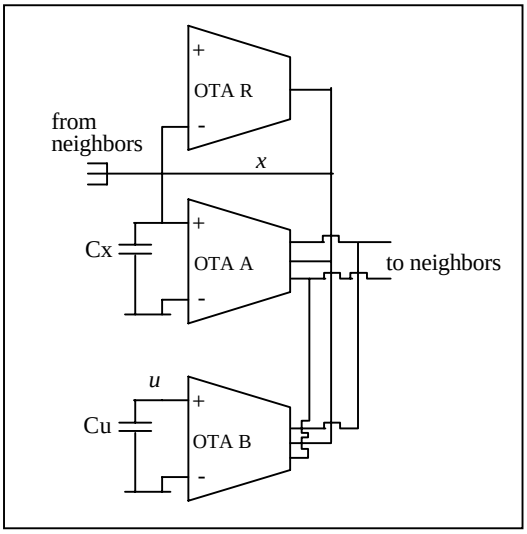


Figure 16 Cruz & Chua's CNN cell

Each cell contains about 50 transistors and occupies an area of 31,000μm^2 (32 cells per mm^2). Settling time of the circuit is of order 1μs.

Halonen, Porra, Roska and Chua[16] realized in similar way a CNN containing local digital memory and logic. Weights are programmable in a discrete set, and biases continuously; logic operations may be performed on steady outputs, and they can be fed back to cell input. In a 4×4 prototype, realized in 2μm CMOS technology, each cell contains about 500 transistor and occupies 1mm^2. Settling time is about 3μs.

Nossek, Seiler, Roska and Chua[8] propose a fully-programmable scheme based on operational amplifiers (op amp) and variable conductance blocks. The inner cell circuit is first transformed by adding an op amp, so that currents are drawn from virtual ground node instead of inner node, thereby stabilizing its voltage against loading effects (figure 17).

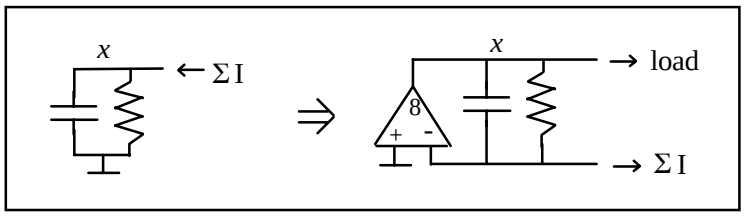


Figure 17 Equivalent inner circuit structures

The circuit is then transformed to a balanced structure (figure 18) that employs variable conductance blocks, that can be realized with four transistors, as in figure 19.

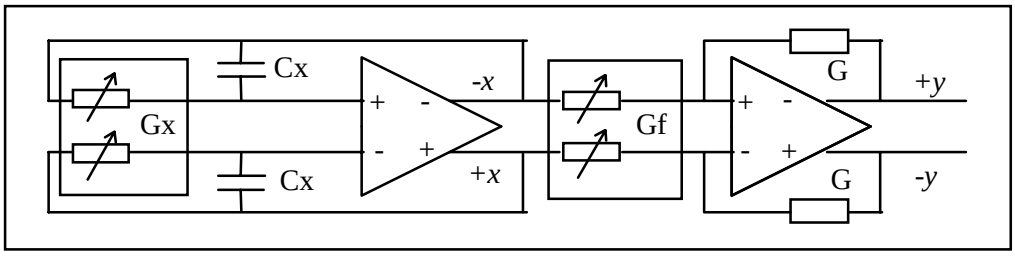


Figure 18 Balanced inner circuit structure

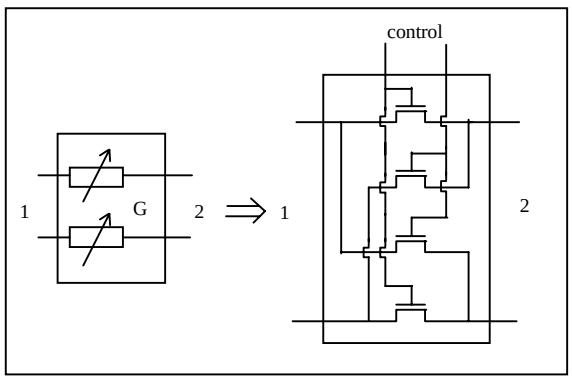


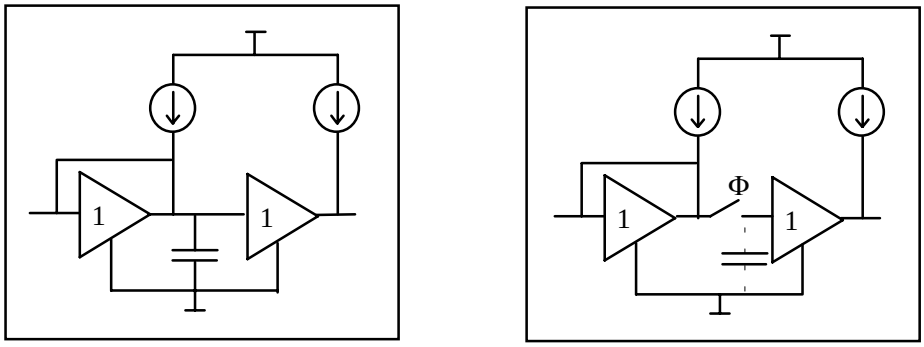
Figure 19 Balanced variable conductance block

Almost the same structure was used by Harrer, Nossek and Stelzl[68] to design a DT-CNN. A 4×4 prototype on hexagonal grid was fabricated in 1.5μm CMOS technology, with 12 cells per mm^2. The circuit operated correctly at 1MHz clock frequency.

A different approach was taken by Rodríguez-Vázquez *et al.*[4], who associate all variables to currents. The reason to abandon voltage variables is that combination of voltage and current variables complicates design for the necessity of scaling signals to compensate nonlinearities, and requires high impedance internal nodes, that cause large time constants to appear. Besides, an efficient way of realizing input is by means of photosensors, that give current output.

Their design is based on the FR-CNN model, obtaining simplified design, good speed/power ratio and low cell complexity. Basic building blocks for this realization are current mirrors, which are used for weighted state replication for connections, and to

realize lossy integration and delay operators required to generate continuous- or discrete-time dynamics (figure 20)



(a) **(b)**

Figure 20 (a) lossy integrator; (b) half-clock delay in current mode

The quoted paper also reports about realization and testing in 1.6μm CMOS technology of several CT and DT prototypes. 9×9 and 1×16 CT prototypes have less than 20 transistors per cell, so that 60 to 160 cells per mm^2 density is achieved. These circuits settle in about 0.25 to 1.5μs. A DT-CNN programmable 9×9 network with local logic was successfully tested at 5 MHz clock frequency. Due to programming and testing circuitry and lines, cells are large: 500μm×500μm.

Important issues to be confronted in practical realizations are those of control and initialization. Accessing all cells at once is generally impossible by means of electronic signals because of the excessive number of lines required. Multiplexed accessing (e.g. by rows) is therefore necessary, together with analog storage, that may be done by capacitors connected to voltage followers. Even bigger difficulty is involved in weight programming. The easiest case is when cloning templates are used, and programming is only allowed in discrete values, that can be selected by a few global lines and some local logic and memory. A promising alternative, especially for image processing purposes, is using on-chip photosensors as input devices.

Realization of DT-CNNs can was also attempted by use of optical devices[69]. Main advantages are speed-of-light computing in the forward path, and possibility of large neighborhood; however, bottlenecks occur in electronic addressing of cells for input (but optical addressing might also be implemented) and in electronic feedback of intermediate results.

Main building blocks for the optical realization are liquid crystal devices (Spatial Light Modulators - SLM), and lenses. SLMs are used to perform analog multiplication between images, by applying variable attenuation to an image transmitted through the panel; their nonlinear sigmoid-like transparency vs. voltage characteristic is also used to implement output nonlinearity. Lenses are used to realize cross-correlation with cloning template. In fact, they realize a Fourier transform, which can be followed by SLM multiplication by a hologram representing cloning template. Inverse transform is obtained by another lens, after observing that a second direct transform yields a mirror image of the desired inverse transform. Complete optical CNN block scheme is depicted in figure 21.

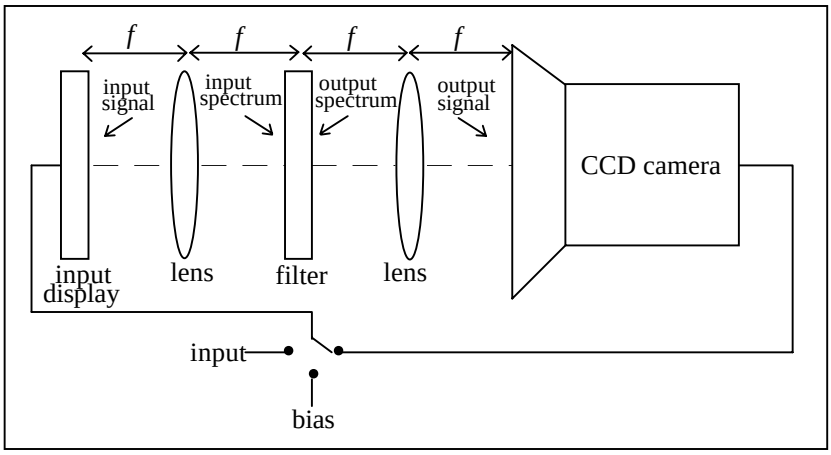


Figure 21 Optical realization of a DT-CNN.

In order to minimize speed limitations of optical CNNs, design should concentrate on minimizing the number of time steps necessary to obtain desired task, by maximizing derivatives of state components and enlarging neighborhood size[70].

7. Appendix

Table of symbols

symbol	*meaning*	*defin.*
A,B	feedback and input coefficients or functions	2.2.1
$\hat{A},\ \hat{B}$	nonlinear feedback and input functions	2.3.3
$A^{\tau},\ B^{\tau}$	nonlinear feedback and input functions	2.3.3
$\boldsymbol{A},\boldsymbol{B}$	cloning template matrices	2.2.2
$\tilde{\boldsymbol{A}},\ \tilde{\boldsymbol{B}}$	feedback and input coefficient matrices	3
$\mathcal{A},\ \mathcal{B}$	feedback, input functionals	2.2.1
$\mathcal{CY}$	cell grid	2.1.1
d	distance function defined over $\mathcal{CY}$	2.1.1
Δ_t	generic differential operator in time domain	2.2.1
f	cell output function	2.1.1
$\mathcal{F}$	cell output functional	2.1.1
g	local feedback function	2.1.1
$\boldsymbol{i}$	imaginary unit	
I	cell bias	
$\mathcal{I}$	domain of indices	2.1.1

$\mathfrak{I}$	set of natural numbers	
m	discrete time delay	2.1.1
μ_i,ν_i	indices of feedback and input functionals	2.1.1
n	discrete time	2.1.1
$N_r(\bullet)$	neighborhood function	2.1.1
$\mathbf{p}^A,\ \mathbf{p}^B$	parameters of feedback and input functional	2.1.1
r	neighborhood size	2.1.1
$\mathfrak{R}$	set of real numbers	
σ	unit step function	2.3.7
t	continuous time	2.1.1
τ	continuous time delay	2.1.1
u	cell input	2.1.1
x	cell state	2.1.1
y	cell output	2.1.1

Table of acronyms

acronym	*meaning*	*defin.*
BSB	Brain-State-in-a-Box	
CA	Cellular Automaton	
CAM	Content-Addressable Memory	
CCD	Connected Component Detector	1
CH-NN	Continuous Hopfield NN	
CNN	Cellular Neural Network	2.1; 2.2
CNNL	CNN with Local Logic	2.3.10
CT-CNN	Continuous Time CNN	2.3
CY-CNN	Chua & Yang's CNN	2.3.1
D-CNN	Delay CNN	2.3.4
DCT-CNN	Delay-type Cloning Template CNN	2.3.4
DT-CNN	Discrete Time CNN	2.3.12
FF-CNN	Feed-Forward CNN	2.3.5
FR-CNN	Full Range CNN	2.3.3
LCT-CNN	Linear Cloning Template CNN	2.3.2
LT-CNN	Linear Threshold CNN	2.3.5
MLP	Multi-Layer Perceptron	
MNS-CNN	Multiple Neighborhood Size CNN	2.3.7
MODA	Moving Object Detecting Architecture	2.3.8
MV-CNN	Multi-Valued CNN	2.3.13
NL-CNN	Non-Linear CNN	2.3.4
NLCT-CNN	Non-Linear Cloning Template CNN	2.3.4
NN	Neural Network	
NUP-CNN	Non-Uniform Processor CNN	2.3.7
OTA	Operational Transconductance Amplifier	
PCL	Positive Cell-Linking	3
P-CNN	Polynomial CNN	2.3.6
PDE	Partial Differential Equation	
PWL	Piece-Wise Linear	
RO-CNN	Relaxation Oscillator CNN	2.3.9
SA	Systolic Array	
SLM	Spatial Light Modulator	6

8. References

In this section, the following abbreviations are used:
CNNA-90: Proc. of IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA-90), Budapest, Hungary, Dec. 16-19, 1990
CNNA-92: Proc. of Second IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA-92), Munich, Germany, October 14-16, 1992
Only most recent versions of papers on the same topic have been quoted.

1. K.R. Crounse, T. Roska, L.O. Chua, "Image Halftoning with CNNs", *IEEE Trans. on Circ. and Syst.* **CAS-II-40**(3) (1993)
2. L.O. Chua, L. Yang, "Cellular Neural Networks: Theory", *IEEE Trans. on Circ. and Syst.* **CAS-35**(10), 1257 (1988)
3. L.O. Chua, T. Roska, "The CNN Universal Machine - Part 1: The Architecture", *CNNA-92*, 1
4. A. Rodríguez-Vázquez, S. Espejo, R. Domínguez-Castro, J.L. Huertas, E. Sánchez-Sinencio, "Current-Mode Techniques for the Implementation of Continuous and Discrete-Time Cellular Neural Networks", *IEEE Trans. on Circ. and Syst.* **CAS-II-40**(3) (1993)
5. J. Anderson, J. Silverstein, S. Ritz, R. Jones, "Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model", *Psychological Review*, **84**, 413 (1977)
6. T. Roska, L.O. Chua, "Cellular Neural Networks with Non-linear and Delay-Type Template Elements and Non-Uniform Grids", *Int. j. c. th. appl.* **20**, 469 (1992)
7. L.O. Chua, B. Shi, "Exploiting Cellular Automata in the Design of Cellular Neural Networks for Binary Image Processing", *University of California at Berkeley, memorandum no.* UCB/ERL M89/130 (1989)
8. J.A. Nossek, G. Seiler, T. Roska, L.O. Chua, "Cellular Neural Networks: Theory and Circuit Design", *Int. j. c. th. appl*. **20**, 533 (1992)
9. A. Barone, M. Balsi, V. Cimagalli, "Polynomial Cellular Neural Network: A New Dynamical Circuit for Pattern Recognition", *Proc. of Int. Specialist Workshop on "Nonlinear Dynamics of Electronic Systems", Dresden, Germany, July 23-24, 1993*
10. J. Henseler, P.J. Braspenning, "A Cortex-like Architecture of a Cellular Neural Network", *CNNA-90*, 244
11. S. Tan, J. Hao, J. Vandewalle, "Cellular Neural Networks as a Model of Associative Memories", *CNNA-90*, 27
12. V. Cimagalli, M. Bobbi, M. Balsi, "MODA: Moving Object Detecting Architecture", *IEEE Trans. on Circ. and Syst.* **CAS-II-40**(3) (1993)
13 J. Henseler, P.J. Braspenning, "Membrain: A Cellular Neural Network Model Based on a Vibrating Membrane", *Int. j. c. th. appl.* **20**, 483 (1992)
14. A. Barone, M. Balsi, V. Cimagalli, "Cellular Networks of Oscillators", *CNNA-92*, 246
15. T. Roska, L.O. Chua, "The Dual CNN Analog Software", *Hungarian Academy of Sciences, Budapest, Hungary, report no.* DNS-2-1992

16. K. Halonen, V. Porra, T. Roska, L.O. Chua, "Programmable Analog VLSI Chip with Local Digital Logic", *Int. j. circ. th. appl.*, **20**, 573 (1992)
17. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties like Those of Two-State Neurons", *Proc. Nat. Acad. Sci*, **81**, 3088 (1984)
18. H. Harrer, J.A. Nossek, "Discrete-time Cellular Neural Networks", *Int. j. c. th. appl.* **20**, 453 (1992)
19. L.O Chua, T. Roska, P.L. Venetianer, A. Zaràndy, "Some Novel Capabilities of CNN: Game of Life and Examples of Multipath Algorithms", *Hungarian Academy of Sciences, Budapest, Hungary, report no.* DNS-3-1992
20. N.N. Aizemberg, I.N. Aizemberg, "CNN Based on Multi-Valued Neuron as a Model of Associative Memory for Grey-Scale Images", *CNNA-92*, 37
21. S. Grossberg, "Some Nonlinear Networks Capable of Learning a Spatial Pattern of Arbitrary Complexity", *Proc. Nat. Acad. Sci.*, **59**, 368 (1968)
22. S.-I. Amari, "Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements", *IEEE Trans. on Computers*, **C-21**, 1197 (1972)
23. T. Roska, L.O. Chua, "Cellular Neural Networks with Non-linear and Delay-Type Template Elements", *CNNA-90*, 12
24. T. Roska, C.W. Wu, M. Balsi, L.O. Chua, "Stability and Dynamics of Delay-Type General and Cellular Neural Networks", *IEEE Trans. on Circ. and Syst.* **CAS-I-39**(6), 487 (1992)
25. T. Roska, C.W. Wu, M. Balsi, L.O. Chua, "Stability of CNNs with Dominant Nonlinear and Delay-Type Templates", *IEEE Trans. on Circ. and Syst.* **CAS-I-40**(3) (1993)
26. L.O Chua, T. Roska, "Stability of a Class of Nonreciprocal Cellular Neural Networks", *IEEE Trans. on Circ. and Syst.* **CAS-37**(12), 1520 (1990)
27. F. Zou and J.A. Nossek, "Stability of Cellular Neural Networks with Opposite-Sign Templates", *IEEE Trans. on Circ. and Syst.* **CAS-38**, 675 (1991)
28. M. Balsi, "Stability of Cellular Neural Networks with One-Dimensional Templates", *Int. j. circ. th. appl., in press*
29. L.O. Chua, C.W. Wu, "On the Universe of Stable Cellular Neural Networks", *Int. j. c. th. appl.* **20**, 497 (1992)
30. M. Balsi, "Recurrent Back-Propagation for Cellular Neural Networks", *Proc. of European Conference on Circuit Theory and Design (ECCTD'93), Davos, Switzerland, Aug.30-Sep. 3, 1993 (Elsevier)*
31. M. Balsi, "Generalized CNN: Potentials of a CNN with Non-Uniform Weights", *CNNA-92*, 129
32. P.P. Civalleri, M. Gilli, "Some Stability Properties of CNNs with Delay", *CNNA-92*, 94
33. L.O. Chua, L. Yang, "Cellular Neural Networks: Applications", *IEEE Trans. on Circ. and Syst.* **CAS-35**(10), 1272 (1988)
34. A. Dzielinski, S. Skoneczny, R. Zbikowski, S. Kuklinski, "Cellular Neural Network Application to Moiré Pattern Filtering", *CNNA-90*, 139

35. T: Matsumoto, T. Yokohama, H. Suzuki, R. Furukawa, A. Oshimoto, T. Shimmi, Y. Matsushita, T. Seo, L.O. Chua, "Several Image Processing Examples by CNN", *CNNA-90*, 100
36. D.-H. Yu, C.-Y. Ho, X. Yu, S. Mori, "On the Application of Cellular Automata to Image Thinning with Cellular Neural Networks", *CNNA-92*, 210
37. G. Seiler, "Small Object Counting with Cellular Neural Networks", *CNNA-90,* 114
38. M. Tanaka, K.R. Crounse, T. Roska, "Template Synthesis of Cellular Neural Networks for Information Coding and Decoding", *CNNA-92*, 29
39. T. Szirànyi, T. Roska, P. Szolgay, "A Cellular Neural Network Embedded in a a Dual Computing Structure (CNND) and its Use for Character Recognition", *CNNA-90*, 92
40. K. Nakayama, Y. Chigawa, "Japanese Kanji Character Recognition Using Cellular Neural Networks and Modified Self-Organizing Feature Map", *CNNA-92*, 191
41. P. Szolgay, I. Kispàl, T. Kozek, "An Experimental System for Optical Detection of Layout Errors of Printed Circuit Boards Using Learned CNN Templates", *CNNA-92*, 203
42. Gy. Eröss, T. Boros, À. Kiss, A. Radvànyi, T. Roska, J. Bitò, J. Vass, "Optical Tracking System for Automatic Guided Vehicles Using Cellular Neural Networks", *CNNA-92*, 216
43. T. Boros, K. Lotz, A. Radvànyi, T. Roska, "Some Useful, New, Nonlinear and Delay-Type Templates", *Hungarian Academy of Sciences, Budapest, Hungary, report no. DNS-1-1991*
44. S. Jankowski, R. Wanczuk, "Nonlinear CNN Cloning Templates for Image Thicking", *CNNA-92*, 197
45. P. Nachbar, A.J. Schuler, T. Füssl, J.A. Nossek, L.O. Chua, "Robustness of Attractor Networks and an Improved Convex Corner Detector", *CNNA-92*, 55
46. S. Paul, K. Hüper, J.A. Nossek, L.O. Chua, "Mapping Nonlinear Lattice Equations onto CNNs", *IEEE Trans. on Circ. and Syst.* **CAS-I-40**(3) (1993)
47. T. Roska, K. Lotz, J. Hàmori, E. Làbos, J. Takàcs, "The CNN Model in the Visual Pathway - Part I: the CNN-Retina and some Direction- and Length-Selective Mechanisms", *Hungarian Academy of Sciences, Budapest, Hungary, report no. DNS-8-1991*
48. Z. Galias, "Designing Discrete-Time Cellular Neural Networks for the Evaluation of Local Boolean Functions", *CNNA-92*, 23
49. T. Roska, T. Boros, A. Radvànyi, P. Thiran, L.O. Chua, "Detecting Moving and Standing Objects Using Cellular Neural Networks", *Int. j. circ. th. appl.,* **20**(5), 613 (1992)
50. C. Güzelis, "Supervised Learning of the Steady-State Outputs in Generalized Cellular Neural Networks", *CNNA-92,* 74
51. A. Radvànyi, "A Dual CNN Model of Cyclopean Perception and its Application Potentials in Artificial Stereopsis", *CNNA-92*, 222
52. Y. Andreyev, Y. Belsky, A. Dmitriev and D. Kuminov, "Inhomogeneous Cellular Neural Networks: Possibility of Functional Device Design", *CNNA-92*, 135

53. G. Seiler, A.J. Schuler, J.A. Nossek, "Design of Robust Cellular Neural Networks", *Technische Universität München, Munich, Germany, report no. TUM-LNS-TR-91-13* (1991)
54. G. Martinelli, R. Perfetti, "Associative Memory Design Using Space-Varying Cellular Neural Networks", *CNNA-92*, 117
55. H. Mizutani, "The Effective Weighting Method of CNN as a Recall", CNNA-92, 86
56. H. Harrer, J.A. Nossek, F. Zou, "A Learning Algorithm for Discrete-Time Cellular Neural Networks", *Technische Universität München, Munich, Germany, report no. TUM-LNS-TR-91-1* (1991)
57. L.O. Chua, P. Thiran, "An Analythic Method for Designing Simple Cellular Neural Networks", *IEEE Trans. on Circ. and Syst.* **CAS-38**(11), 1332 (1991)
58. J.A. Osuna, G.S. Moschytz, "Exact Design of Reciprocal Cellular Neural Networks", *CNNA-92*, 11
59. K. Slot, "Cellular Neural Networks Design for Solving Specific Image-Processing Problems", *Int. j. circ. th. appl.* **20**, 629 (1992)
60. F. Zou, S. Schwarz, J.A. Nossek, "Cellular Neural Network Design Using a Learning Algorithm"; *CNNA-90*, 73
61. P. Szolgay, T. Kozek, "Optical Detection of Layout Errors of Printed Circuit Boards Using Learned CNN Templates", *Hungarian Academy of Sciences, Budapest, Hungary, report no. DNS-10-1991*
62. A.J. Schuler, P. Nachbar, J.A. Nossek, L.O. Chua, "Learning State Space Trajectories in Cellular Neural Networks", *CNNA-92*, 68
63. T. Kozek, T. Roska, L.O. Chua, "Genetic Algorithm for CNN Template Design", *IEEE Trans. on Circ. and Syst.* **CAS-I-40**(3) (1993)
64. C. Güzelis, "Supervised Learning of the Steady-State Outputs in Generalized Cellular Neural Networks", *CNNA-92*, 74
65. L.K. Hansen, "Bolzmann Learning of Parameters in Cellular Neural Networks", *CNNA-92*, 62
66. H. Magnussen, J.A. Nossek, "Towards a Learning Algorithm for Discrete-Time Cellular Neural Networks", *CNNA-92*, 80
67. J.M. Cruz, L.O. Chua, "Design of High-Speed, High-Density CNNs in CMOS Technology", *Int, j. circ. th. appl.*, **20**, 555 (1992
68. H. Harrer, J.A. Nossek, R. Stelzl, "An Analog Implementation of Discrete-Time Cellular Neural Networks", *IEEE Trans. on Neural Networks* **NN-3**(3), 466 (1992)
69. E. Lüder, N. Frühauf, "Optical Signal Processing for CNN's", *CNNA-92*, 45
70. K. Slot, "Optically Realized Feedback Cellular Neural Networks", *CNNA-92*, 175