
*Neural Processing Letters* **16:** 111–120, 2002.



# Gradient Computation of Continuous-Time Cellular Neural/Nonlinear Networks with Linear Templates via the CNN Universal Machine

MÁTYÁS BRENDEL, TAMÁS ROSKA and GUSZTÁV BÁRTFAI
*Analogic and Neural Computing Systems Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, P.O.B. 63. H-1502 Budapest, Hungary.*
*e-mail: roska@sztaki.hu*

**Abstract.** Single-layer, continuous-time cellular neural/nonlinear networks (CNN) are considered with linear templates. The networks are programmed by the template-parameters. A fundamental question in template training or adaptation is the gradient computation or approximation of the error as a function of the template parameters. Exact equations are developed for computing the gradients. These equations are similar to the CNN network equations, i.e. they have the same neighborhood and connectivity as the original CNN network. It is shown that a CNN network, with a modified output function, can compute the gradients. Thus, fast on-line gradient computation is possible via the CNN Universal Machine, which allows on-line adaptation and training. The method for computing the gradient on-chip is investigated and demonstrated.

**Key words.** cellular neural networks, gradient-method, learning, training

## 1. Introduction

The design and training of neural networks are important and challenging subjects of neurocomputing sciences. Since the introduction of cellular neural networks [1], the design and learning of cellular neural networks (CNN) have also been major topics of research. A summary of template design methods can be found in [2] (see [3, 4] as well). Template design is most developed for (i) feed-forward, (ii) uncoupled and (iii) coupled, binary input and binary output templates. The study of CNN template training (i.e. CNN learning) has also been an important topic ever since the first CNN conference CNNA'90 [7]. A comprehensive summary of the early period of design and learning can be found in [8, 9].

Template training was always based on basically the same mean square error concept. However, various kinds of minimization methods were used for optimization of the error function over the parameter space. These optimization methods are the gradient methods [10]; the evolutionary methods (see [11, 12]) and the statistical optimization methods [13].

This work focuses on gradient optimization methods and on-chip CNN learning. This means that the gradients are to be computed via CNN hardware. In [14]

the methods used for computing or estimating gradients of neural networks are summarized (for example, backpropagation-through-time and truncation). In general, it can be stated that using exact gradients allows a wider range of templates to be trained, but places high demand on computational resources as well [10]. On the other hand, some simple templates may be trained by using only estimated gradients. Our experiments have also shown that the so-called propagation templates usually cannot be trained by truncated gradients. Consequently, our goal is to compute complete, precise gradients.

In the case of pure feed-forward networks gradient computation can be solved with back-propagation, which has been known from general neural network theory for a long time. For recurrent networks recurrent back-propagation was introduced, which applies to CNN as well. A general approach is network reciprocity ([16–18]), which automates the diagrammatic derivation of the reciprocal network, computing the gradients of the original network for arbitrary network structures.

In [15] a unified approach was presented for computating precise gradients of multilayer continuous-time CNNs (CT-CNN).

The main drawback of computing complete gradients is the huge demand on resources, which makes this method intractable for practical problems. The main contribution of our work is to give a solution for the computation of complete gradients, which can be used for practical applications. The idea introduced here is to show that the reciprocal networks of a CNN are also recurrent finite neighborhood networks with the same neighborhood and order as the original network, and hence they can be computed with the CNN hardware itself. This fact resembles the reciprocal network approach (see [16–18]), however, we use algebraic derivation (chain rule expansion). By our approach the high demand on computational resources is fulfilled by a fast hardware solution.

The significance of this possibility is underlined by the fact that in an analogic CNN computer, the CNN Universal Machine (see [19, 20]) stored program can be defined to calculate the gradients via the same hardware. Once we are able to calculate the gradients various architectural possibilities for adaptation and plasticity are available (see [27–29]) to use this information.

## 2. CNN Model

Our paper deals with the continuous-time cellular neural network model with linear templates defined as follows:

$$\begin{aligned}\frac{\mathrm{d}x_{i,j}(t)}{\mathrm{d}t} &= -x_{i,j}(t) + \sum_{k,l\in S(i,j)} A(k-i,j-l)y_{k,l}(t) + \\ &\quad + \sum_{k,l\in S(i,j)} B(k-i,j-l)u_{i,j}(t) + z \\ y_{i,j}(t) &= f(x_{i,j}(t))\end{aligned} \tag{1}$$

where $x_{i,j}$ is the state, $u_{i,j}$ is the input and $y_{i,j}$ is the output of the neuron at position $(i,j)$, $A$ is the feedback and $B$ is the feed-forward template, $z$ is the bias and $f$ is the output function. $S_r(i,j)$ is the sphere of influence of $(i,j)$. $S(i,j) = \{(k,l) : \max(|k-i|, |j-l| \leqslant r\})$. The first equation is called network equation, and the second output equation. The described CT-CNN is commonly used for image processing tasks, where the input image is loaded to the input $U = [u_{i,j}]$ and/or to the initial state $X_0 = [x_{i,j}(0)]$, and the output image is taken at a certain time $T$ from the output of the network $Y = [y_{i,j}(T)]$.

Various image-processing tasks may be solved by choosing appropriate feedback (A), feed-forward (B) templates and bias (z). The program of the CT-CNN is actually this set of parameters $P = A \cup B \cup z$. One particular parameter will be denoted by $p \in P$.

For an arbitrary parameter-set P an error function may be defined by considering the specific image-processing task. For example, the average mean-square error for a predefined image is as follows:

$$E(P) = \frac{1}{2a} \sum_{i,j} (y_{i,j}(T) - d_{i,j})^2 \tag{2}$$

where $a$ is the area of the picture, $y_{i,j}(T)$ is the computed output, using the parameter $P$, and $d_{i,j}$ is the value of the desired output at the position $(i,j)$.

For training or other purposes the partial derivatives of the error related to the parameters are fundamental:

$$\frac{\mathrm{d}E(P)}{\mathrm{d}p} = \frac{1}{a} \sum_{i,j} (y_{i,j}(T) - d_{i,j}) \frac{\mathrm{d}y_{i,j}(T)}{\mathrm{d}p} \tag{3}$$

## 3. Computation of the Gradients

First some notations are introduced for the partial derivatives:

$$\delta p y_{i,j}(t) = \frac{\mathrm{d}y_{i,j}(t)}{\mathrm{d}p}, \quad \delta p x_{i,j}(t) = \frac{\mathrm{d}x_{i,j}(t)}{\mathrm{d}p},$$

where $p$ shall be replaced by a specific parameter. These two variables are the output and state of the reciprocal network. It is obvious that the following equation holds every time, due to the output equation of (1):

$$\delta y p_{i,j}(t) = \frac{\partial f}{\partial x}(x_{i,j}(t)) \delta x p_{i,j}(t) \tag{4}$$

In order to compute the gradients the network equation of (1) has to be considered. First, the derivative of the network equation has to be taken with respect to the variable $p$. Second the resulting equation has to be solved in an initial-state problem by integration up to the time $T$. The form of the resulting equation, which has to be integrated depends on the kind of parameter $p$.

If $p = A(m,n)$ then the derivative of the network equation has the form of:

$$\frac{\mathrm{d}^2 x_{i,j}(t)}{\mathrm{d}t\mathrm{d}p} = -\frac{\mathrm{d}x_{i,j}(t)}{\mathrm{d}p} + \sum_{(k,l)\in S(i,j)} \frac{\mathrm{d}A(k-i,j-l)}{\mathrm{d}p} y_{k,l}(t) + \\ + \sum_{(k,l)\in S(i,j)} A(k-i,j-l)\frac{y_{k,l}(t)}{\mathrm{d}p} + \\ + \frac{\mathrm{d}}{\mathrm{d}p}\left( \sum_{(k,l)\in S(i,j)} B(k-i,j-l)u_{k,l}(t) + z \right) \tag{5}$$

where the first sum reduces to one term, and the last term is 0. With the appropriate substitutions the equation reduces to the form of:

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta p x_{i,j}(t) = -\delta p x_{i,j}(t) + \sum_{k,l\in S(i,j)} A(k-i,j-l)\delta p y_{k,l}(t) + y_{i+m,j+n}(t) \tag{6}$$

This means that the templates of the reciprocal CNN are $A' = A$, $B' = 1$, $z' = 0$ and the input is the original output. If $p = B(m,n)$ the first sum in equation (5) is 0, and the last sum is $u_{i+m,j+n}(t)$, thus the equation becomes of the following form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta p x_{i,j}(t) = -\delta p x_{i,j}(t) + \sum_{k,l\in S(i,j)} A(k-i,j-l)\delta p y_{k,l}(t) + u_{i+m,j+n}(t) \tag{7}$$

This means that the templates of the reciprocal CNN are $A' = A$, $B' = 1$, $z' = 0$ and the input of this network is the same as the input of the original network. Finally if $p = z$ then in equation (5) the first sum is 0, and the last sum is equal to 1, thus the equation reduces to the following form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta p x_{i,j}(t) = -\delta p x_{i,j}(t) + \sum_{k,l\in S(i,j)} A(k-i,j-l)\delta p y_{k,l}(t) + 1 \tag{8}$$

This means that the templates of the reciprocal CNN are $A' = A$, $B' = 0$, $z' = 1$ and there is no input. The gradients have to be computed by integrating these equations using zero initial state $\delta p x_{i,j}(0) = 0$. Finally $\delta p y_{i,j}(T)$ has to be taken and substituted into (3).

## 4. Hardware Solution

Equations (6), (7) or (8) can be considered as special CNN equations if $\delta px$ is the state and $\delta py$ is the output of the cell and $y$ (the output of the original network) is the input.

The only problem is the special output function in (4). If this output function can be realized in the hardware, gradient computation can be carried out with the CNN itself (see Figure 1).

If separate CNN chips are available for all the partial derivatives which have to be computed, then the gradient computation can be solved by using the reciprocal templates and by connecting the chips. The output and state of the original CNN

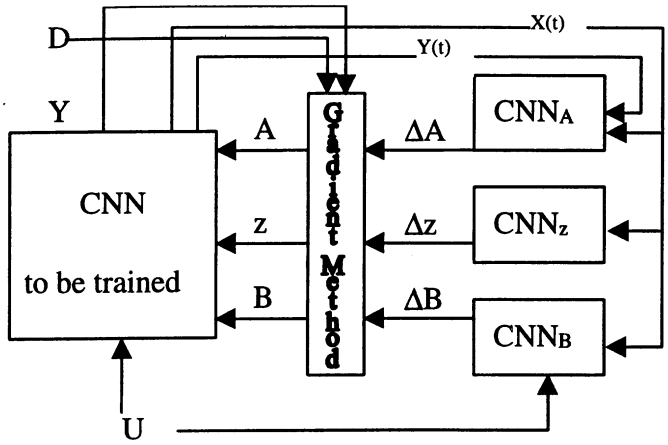


*Figure 1.* Flowchart of on-line gradient computation via CNN and the reverse CNNs. The reverse CNNs computing the gradients related to the different types of template parameters are denoted as $CNN_A$, $CNN_B$, $CNN_z$ respectively.

shall be fed into the reverse networks, which compute the partial derivatives. The time demand of gradient computation is the same for one template parameter as for the computation of the original CNN.

If only one CNN chip is available, the gradient can be approximated by using the CNN Universal Machine. The state and the output of the original CNN shall be sampled and stored appropriately for approximating the state and the output as a function in time.

In both cases on-line and fast gradient computation is possible, which is important for training or adaptation purposes.

In both cases the computation of the gradient of one parameter consumes exactly the same time, as the computation of the original template. This is $T\tau_{CNN}$ where $T$ is involved as a free parameter in the description of the algorithm, i.e. dependent on task to be trained, and $\tau_{CNN}$ is a physical parameter of the chip. This value vas for example 250 ns for the 'Seville'-chip [30].

## 5. Tests and Results

Since only one chip was available, therefore the second possibility was tested on chip. The following diffusion template was used:

$$A = \begin{array}{|c|c|c|}\hline 0.1 & 1 & 0.1 \\ \hline 0.1 & 0 & 0.1 \\ \hline 0.1 & 0.1 & 0.1 \\ \hline \end{array} \qquad B = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad z = \boxed{0}$$

The center element of the template $A$ was modified to $p = a_{00} = -0.1$. The gradient for this parameter was the question of the experiment. The advantage of using diffusion is that there is no saturation, and all cells are in the linear domain. Therefore

the term $\delta f/\delta x_{ij}(x_{ij}) = 1$, and thus in equation (4) $\delta f/\delta x_{ij}(x_{ij}(t))\ \delta xp_{ij}(t) = \delta xp_{ij}(t)$, which is $f(\delta xp_{ij}(t))$ assuming no saturation. In this case the gradient computation can be established on the current architecture. To avoid saturation the values can be scaled down if needed.

The transient of $T = 1\tau_{\mathrm{CNN}}$ was sampled in 10 discrete time steps. This gives an approximation of the changing output, and this was used as the input of the reverse CNN computing the gradient. This means that $y(t)$ was approximated with the series $y(0), y(0.1), \ldots, y(0.9)$. The gradient was computed by simulation and on chip as well, with the same method presented above. The result is compared in Figure 2.

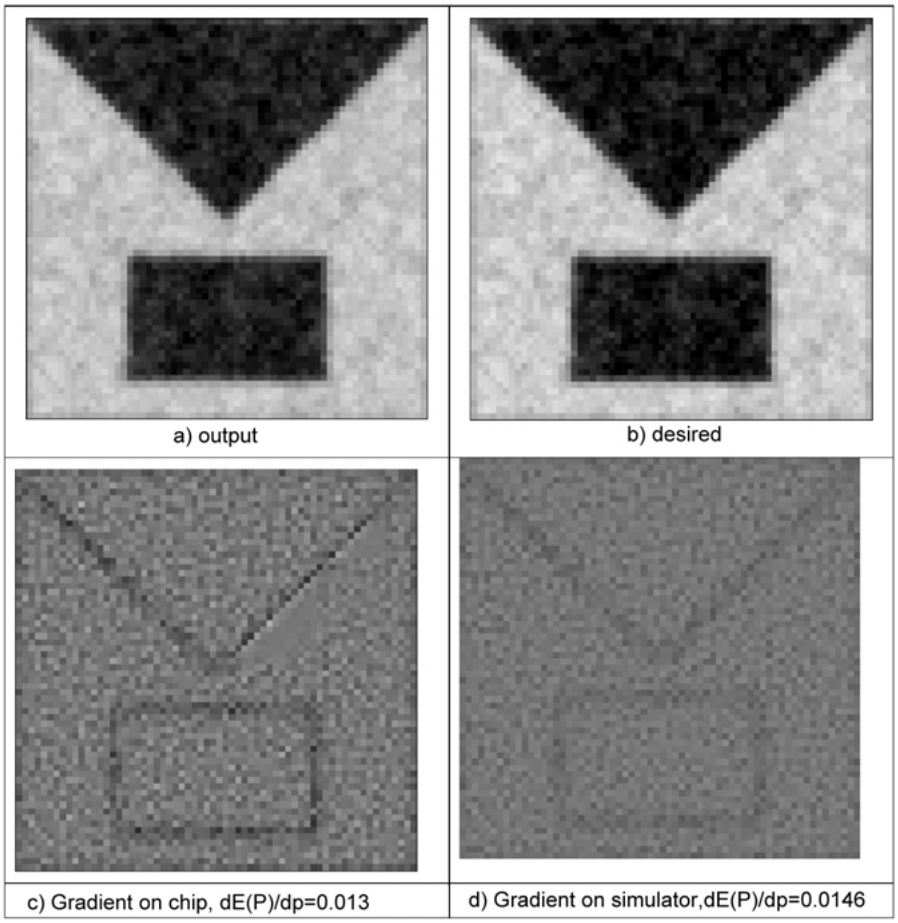


*Figure 2.* a) Result of the modified template, b) desired result with the desired template, c) result of computation on-chip, d) result of the approximation of the gradient on chip. The pictures show the matrix $[(y(T)_{ij} - d_{ij})dy_{ij} - (T)/dp_{ij}]_{ij}$. Both gradients are sufficient and comparable. Regrettably there are also fault on the result with chip. The overall contrast is larger on-chip, which can be tuned, if it is important. The gray blur on the right bottom of the black triangle in picture c is however presumably a failure of the used chip. The colors code values with a linear function on the ranges $[-1,1] \rightarrow$[white,black].

We cannot determine the real gradient, since the close formula of the integral of equation (6) is not known. Therefore we assume that the simulation approximates a good gradient and we compare the chip-result to this value. The computed gradient 0.013 is appropriate, since the center element should be increased to achieve the desired template. As it can be seen on the image, the computation on-chip is not perfect, but the result is suitable.

## 6. Discussion

In the following, we review how our results fit into a broader area of research. There are various methods of optimization that are used in CNN template training (see, for example [11, 13]). Evolutionary and statistical methods are robust but they are slow since they require a large number of epochs to explore the solution space to find good directions. Moreover, these methods – although work for a wide range of templates – require external (i.e. off-CNN) mechanisms typically implemented in software. Therefore, fast and on-line learning is difficult.

Gradient methods were used in [8–15] for example, but there were some difficulties. One problem was that by using the standard output function, the error function is non-continuous, and therefore the continuation method had to be introduced in [23]. The work in [14], summarizes gradient computing methods for a network similar to CNN namely, for the Simultaneous Recurrent Network (SRN). In this work the finite neighborhood case is also discussed, thus this network becomes very similar to a DT-CNN.

The computation of gradients for neural networks is a fundamental topic of research. There exists an approach [16–18] for automatic generation of the so-called reciprocal network for arbitrary (i.e. also recurrent) network structures, which computes the gradients of the original network. This approach can be used for CNNs as well.

The method of computing gradients for a CT-CNN network can be established with the help of using the above mentioned reciprocal network approach, or specifically by the simple, iterative use of the chain rule.

The results presented in this paper can be related to other research on this topic. More specifically, we discuss our contribution in relation to three papers.

The first one is the paper of R. Tetzlaff and D. Wolf [15] which introduces a learning algorithm for the dynamics of CNN with nonlinear templates. It is shown that the gradients for multilayer CT-CNNs can in general, be given with a coupled system of differential equations. In our paper, we extend this result by rigorously deriving exact analytical formulae for the gradients for single-layer CNNs with linear templates and by showing that these gradients can be computed with the CNN itself (i.e., no external mechanism is required for learning). Besides it is necessary to use the continuation method for CNN's with piecewise linear output function. Furthermore, we explicitly address image-processing tasks related to propagating templates.

The second paper is the work of P. Xiaozhong and Paul J. Werbos [14], which basically describes the basics of gradient optimization based training methods of the Simultaneous Recurrent Networks (SRN). The paper describes backpropagation-through-time (BTT) training and gradient computation for SRN's among others. The case of local neighborhood connectivity is also discussed. A SRN with local neighborhood is very similar to a DT-CNN, although the SRN is more complex since it consists of so called composite cells. Our work applies gradient-based training to CNNs and the method of gradient computation is detailed.

The third paper is the work of T. Yang and L. O. Chua [22], which recognizes the possibility of realizing gradient computation for BTT with CNN. However their work is related to the training of SRN's, we address the training of the CNN itself.

## 7. Conclusion

It has been proven that the reciprocal networks of a CNN are recurrent networks with the same connectivity-structure and neighborhood as the original network. Consequently, the computation of gradients, which consumes a prohibitively large amount of resources, can be carried out basically with the same network structure. Thus gradient learning or gradient based parameter adjustment can be done quick and on-line. The time demand of gradient computation with such hardware would be the same for one template parameter as the time demand of the computation of the original CNN itself. This way CNN may become a compact, flexible device without the need of an external device for parameter optimization.

The results are valid for single-layer CNNs with linear templates. However, they can be similarly extended for multilayer CNNs and for nonlinear templates as well. This is a subject of research in the future.

Our experience has also shown that if gradient optimization is used for template training in practice, developing an intelligent and robust gradient algorithm is essential. The development could be based on the methods mentioned in our work.

## Acknowledgement

The advice and encouragement of Professor Paul Werbos are gratefully acknowledged. The research was supported by the grant No. T026555 of the National Research Found of Hungary (OTKA).

## References

1. Chua, L. O. and Yang, L.: Cellular neural networks: theory, *IEEE Trans. on Circuits and Systems,* (CAS), **35** (1988), 1257–1272.
2. Zarándy, Á.: The Art of CNN template design, *Int. J. Circuit Theory and Applications – Special Issue: Theory, Design and Applications of Cellular Neural Networks: Part II: Design and Applications,* (CTA Special Issue - II), **17**(1) (1999), 5–24.

3. Zarándy, Á., Stoffels, A., Roska, T., Werblin, F. and Chua, L. O.: Implementation of binary and grayscale mathematical morphology on the CNN universal machine, *IEEE Trans. Circuits System-I*, **45**(2) (1998), 163–168.
4. Hanggi, M. and Moschytz, G. S.: An exact and direct analytical method for the design of optimally robust CNN templates, *IEEE Trans. on Circuits and Systems I: Special Issue on Bio-Inspired Processors and Cellular Neural Networks for Vision,* (CAS-I Special Issue), **46**(2) (1999), 304–311.
5. Nemes, L., Chua, L. O. and Roska, T.: Implementation of Arbitrary Boolean Functions on the CNN Universal Machine, *Int. J. Circuit Theory and Applications – Special Issue: Theory, Design and Applications of Cellular Neural Networks: Part I: Theory,* (CTA Special Issue - I), **26**(6), guest ed: T. Roska, A. Rodriguez-Vazquez and J. Vandewalle, 1998, pp. 593–610.
6. Crounse, K. R., Fung, E. L. and Chua, L. O.: Efficient implementation of neighborhood logic for cellular neural network universal machine, *IEEE Trans. Circuits System-I*, **44**(4) (1997), 255–361.
7. Zou, F., Schwarz, S. and Nosek, J. A.: Cellular neural network design using a learning algorithm, *Proceedings of the First International Workshop of Cellular Neural Networks and their Applications* (*CNNA-90*), 1990, pp. 73–82.
8. Nossek, J. A.: Design and learning with cellular neural networks, *Proceedings of the Third International Workshop of Cellular Neural Networks and their Applications* (*CNNA-94*), 1994, pp. 137–146.
9. Nossek, J. A.: Design and Learning with Cellular Neural Networks, Microsystems Technology for Multimedia Applications: An Introduction, (Multimedia), Bing Sheu (ed), IEEE Press, 0-7803-1157-4, (1995), 395–404.
10. Pineda, F. J.: Generalization of back-propagation to recurrent neural networks, *Physical Review Letters*, **1** (Nov. 1987), 2229–2232.
11. Kozek, T., Roska, T. and Chua, L. O.: Genetic algorithm for CNN template learning, *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, **40**(6) (1993), 392–402.
12. Szirányi, T. and Csapodi, M.: Texture classification and segmentation by cellular neural network using genetic learning, *Computer Vision and Image Understanding,* (CVIU), **71**(3) (1998), 255–270.
13. Chandler, B., Rekeczky, Cs., Nishio, Y. and Ushida, A.: Adaptive simulated annealing in CNN template learning, *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, **E82-A**, (2), (1999), 398–402.
14. Pang, X. and Werbos, P.: Neural network design for J function approximation in dynamic programming, *Journal on Mathematical Modeling and Scientific Computing* (*Principia Scientia*), *special issue on neural networks,* planned as (1), 1997.
15. Tetzlaff, R. and Wolf, D.: A Learning Algorithm for the Dynamics of CNN with Nonlinear Templates-Part II: Discrete-Time Case, *Proceedings of the Forth International Workshop of Cellular Neural Networks and their Applications* (*CNNA 96*), 1996, pp. 461–466.
16. Wan, E. and Beaufays, F.: Network Reciprocity: A Simple Approach to Derive Gradient Algorithms for Arbitrary Neural Network Structures, *Proc. WCNN'94*, **3**, San Diego, CA, (6–9 June 1994), 382–389.
17. Wan, E. and Beaufays, F.: Diagrammatic derivation of gradient algorithms for neural networks, *Neural Computation*, **8**(1) (1996), 182–201.
18. Beaufays, F. and Wan, E.: A unified approach to derive gradient algorithms for arbitrary neural network structures, *Proc. ICANN'94*, **1**, Sorrento, Italy, (26–29 May 1994), 545–548.
19. Chua, L. O. and Roska, T.: The CNN paradigm, *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, **40**(3) (1993), 147–156.

20. Roska, T. and Chua, L. O.: The CNN universal machine: an analogic array computer, *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, **40**(3) (1993), 163–173.
21. Puffer, F., Tetzlaff, R. and Wolf, D.: A Learning Algorithm for Cellular Neural Networks Solving Partial Differential Equations, *ISSSE 95,* San Francisco, 1995.
22. Yang, T. and Chua, L. O.: Implementing back-propagation-trough-time algorithm using cellular neural networks, *Int. Journ. of Bifurcation and Chaos*, **9**(6) (1999), 1041–1074.
23. Magnussen, H. J. and Nossek, A.: Continuation-based learning algorithm for Discrete-Time Cellular Networks, *Proceedings of the Third International Workshop of Cellular Neural Networks and their Applications (CNNA-94)*, 1994, pp. 171–176.
24. Chua, L. O. and Roska, T.: Cellular Neural Networks: Foundation and primer, Lecture notes for course EE129 at U.C. Berkeley, 1999.
25. Roska, T., Kék, L., Nemes, L., Zarándy, Á., Brendel, M. and Szolgay, P. (eds.): CNN Software Library (Templates and Algorithms) Version 7.2, Research Report (DNS-1-1998), Analogical and Neural Computing Laboratory, MTA SZTAKI, Budapest 1998.
26. Magnussen, H. and Nossek, J. A.: A geometric approach to properties of the discrete-time cellular neural network, *IEEE Trans. on Circuits and Systems I: Fundamental theory and applications*, **41** (1994), 625–634.
27. Roska, T.: Space Variant Adaptive CNN – Fault Tolerance and Plasticity, *Proceedings of International Symposium on Nonlinear Theory and Applications* (*NOLTA'97*), 201–204, Hawaii, 1997.
28. Földesy, P., Kék, L., Roska, T., Zarándy, Á., Roska, T. and Bártfai, G.: Fault Tolerant Design of Analogic CNN Templates and Algorithms - Part I: The Binary Output Case, *IEEE Trans. on Circuits and Systems I: Special Issue on Bio-Inspired Processors and Cellular Neural Networks for Vision*, **46**(2) (1999), 312–322.
29. Roska, T.: Computer-Sensors: spatial-temporal computers for analog array signals, dynamically integrated with sensors, *Journal of VLSI Signal Processing Special Issue: Spatio-temporal Signal Processing with Analogic CNN Visual Microprocessors*, (JVSP Special Issue), **23**(2/3), (November/December 1999), Kluwer 221–238.
30. Espejo, S., Dominguez-Castro, R., Linan, G. and Rodriguez-Vázquez, A.: A 64×64 CNN Universal Chip with Analog and Digital I/O, *Proceedings of 5th IEEE International Conference on Electronics, Circuits and Systems, (ICECS'98)*, 203–206, Lisboa, 1998.