(1)

CNNA'92

PROCEEDINGS

Second International Workshop on Cellular Neural Networks and their Applications

Technical University Munich
Munich, Germany
October 14–16, 1992

IEEE 92TH0498-6

ISBN 0–7803–875–1

Additional copies may be obtained from
IEEE Service Centre
445 Hoes Lane, Piscataway, NJ 08854–1331
IEE catalog No. TH0498-6
ISBN 0–7803–875–1

Welcome to the CNNA '92

Cellular Neural Networks (CNN) are locally interconnected, regularly repeated, analog (continuous- or discrete-time) circuits having a 1–, 2–, or 3–dimensional grid architecture. Each cell in a CNN is a nonlinear dynamic system coupled only to its nearest neighbors. One advantage of CNNs over competing approaches is the ease in implementing VLSI CNN chips, thereby making real time operations possible. Although proposed barely four years ago, numerous applications in the areas of image processing, pattern recognition, robot vision and motion detection have since been reported in several international journals, conferences and in the first IEEE International Workshop on Cellular Neural Networks and their Applications in Budapest in 1990. A special issue of the "International Journal on Circuit Theory and Applications" devoted to CNNs, has already been published this year. Also currently being prepared and expected to appear in March 1993, is a special issue of the "Transactions on Circuits and Systems".

The first symposium proved to be a truly international and quite successful meeting. CNNA '92 has been able to attract even more interesting contributions from all over the world. We are anticipating future exciting and intellectually-stimulating meetings at the forthcoming workshops of this series, which are already being planned for '94 in Roma, and '96 in Sevilla.

This Workshop is intended to share brand-new research results and experiences of the participants, and to explore future research potentials. In support of these objectives, we have made available a CNN workstation equipped with a special hardware accelerator and simulation software in order to provide all participants the opportunity for hands-on experience. This will enable professionals who are not yet active in this new and innovative area, to launch their own research and development on CNN very effectively.

The scientific program comprises a total of 46 contributions including an inaugural lecture by Professor Chua, who introduced CNNs in 1988, and five invited presentations by distinguished experts. The 40 papers, which have been selected from the submissions, are forming six sessions on Theory (I and II), Design, Learning, VLSI-Implementation and Applications. At the end of the workshop an award for the best submitted paper based on an evaluation made by the participants and the scientific committee, will be presented.

We would like to express our sincere thanks to the members of the program committee for their effort in reviewing submitted papers and especially to all authors for submitting their excellent work. Special thanks are also extended to Dr. Gloger and Mrs. Theinert, who bore the main burden of making this workshop organizationally and socially, an enjoyable event for all of us. Finally, the support from the sponsoring organizations (IEEE, VDE/ITG) and especially the generous help from the European Research Office of the US Army is gratefully acknowledged.

We have the pleasure to welcome you to the second International Workshop on Cellular Neural Networks and their Applications (CNNA 92), October 14–16 in Munich, Germany.

We wish you a successful conference.

Munich, September 1992

Josef A. Nossek

Prof. Dr. techn. J.A. Nossek

Scientific Committee

L.O. Chua (UC Berkeley)	**T. Matsumoto** (Waseda Univ.)
V. Cimagalli (Univ. Roma)	**J.A. Nossek** (TU Munich)
K. Halonen (Univ. Helsinki)	**T. Roska** (Hung. Acad. Sci.)
M. Hasler (EPFL Lausanne)	**J. Taylor** (Univ. London)
J. Herault (INP Grenoble)	**J. Vandewalle** (Kath. Univ. Leuven)
HJ. Huertas (Univ. Sevilla)	**H. Wallinga** (Univ. Twente)
E. Lüder (TU Stuttgart)	

Conference Organization

CNNA'92
Institute for Network Theory and Circuit Design
Technical University of Munich, Arcisstr. 21, 8000 Munich 2, GERMANY
Fax: ++49 - 89 - 2105 - 8504, Email: magl@nws.e-technik.tu-muenchen.de
Local Organizing Commttee: Prof. Dr. techn. Josef A. Nossek, (Chairman), Dr. Manfred Gloger, M. Theinert.

Co-Sponsored by

European Research Office of the US Army.
IEEE Region 8 and the IEEE German Section
ITG/VDE

Scientific Program/Table of Contents

Wednesday (October 14, 1992)

9.10 **Inaugural Lecture: The CNN-Universal Machine**
L.O. Chua, University of California Berkeley, USA.
The inaugural lecture includes a demonstration of the CNN workstation by T. Roska, Hungarian Academy of Science. 1

10.10 **Invited Lecture: Biological Implementations of CNNs**
W. Heiligenberg, University of California San Diego, USA -

Session 1 Design
Chairman: H. Wallinga, Univ. of Twente, Netherlands

11.10 **Exact Design of Reciprocal Cellular Neural Networks**
J.A. Osuna, G.S. Moschytz, ETH Zürich, Switzerland 11

11.30 **Cellular Neural Network Design with Continuous Signals**
S. Schwarz, W. Mathis, Univ. of Wuppertal, Germany 17

11.50 **Designing Discrete-Time Cellular Neural Networks for the Evaluation of Local Boolean Functions**
Z. Galias, Univ. of Mining and Metallurgy, Krakow, Poland 23

12.10 **Template Synthesis of Cellular Neural Networks for Information Coding and Decoding**
M. Tanaka, Sophia Univ. Tokyo, Japan; K.R. Crounse, UCB California, USA; T. Roska, Hung. Academy of Sciences, Hungary 29

12.30 **CNN Based on Multi-Valued Neuron as a Model of Associative Memory for Grey-Scale Images**
N.N. Aizenberg, I.N. Aizenberg Univ. of Uzhgorod, Ukraine 36

12.50 **Parameter Tolerances and Generalisation Abilities of Cellular Neural Networks**
O. Kufudaki, M. Novak, Inst. of Computer Science, Prague 42

14.20 **Invited Lecture: Optical Signal Processing for CNNs**
E. Lüder, N. Frühauf, Univ. Stuttgart, Germany 45

Session 2 Learning
Chairman: P.P. Civalleri, Politecnico di Torino, Italy

15.00 **Robustness of Attractor Networks and an Improved Convex Corner Detector**
P. Nachbar, A.J. Schuler, T. Fuessl, J.A. Nossek, TU Munich, Germany, L.O. Chua, Univ. of California Berkely, USA 55

15.20 **Boltzmann Learning of Parameters in Cellular Neural Networks**
L. K. Hansen, Univ. of Denmark, Denmark 62

15.40 **Learning State Space Trajectories in Cellular Neural Networks**
A. Schuler, P. Nachbar, J.A. Nossek, TU Munich, Germany, L.O. Chua, University of California Berkeley, USA. 68

16.30 **Supervised Learning of the Steady-State Outputs in Generalized Cellular Neural Networks**
C. Güzelis, Istanbul Teknik Üniversitesi, Turkey 74

16.50 **Towards a Learning Algorithm for Discrete-Time Cellular Neural Networks**
H. Magnussen, J.A. Nossek, TU Munich, Germany 80

17.10 **The Effective Weighting Method of CNN as a Recall**
H. Mizutani, Shonan Inst. of Technology, Fujisawa, Japan 86

Thursday (October 15, 1992)

9.00 **Invited Lecture : CNNs as Information Processing Devices: Towards a Systematic Approach**
V. Cimagalli, Univ. Roma, Italy 92

Session 3 Theory I
Chairman: M. Hasler, EPFL, Lausanne, Switzerland

9.40 **Some Stability Properties of CNN's with Delay**
P.P. Civalleri, M. Gilli, Politecnico di Torino, Italy 94

10.00 **Two Causes of Instability of Celular Neural Networks**
P. Thiran, EPFL, Lausanne, Switzerland 100

10.20 **Convergence of Reciprocal Time-Discrete Cellular Neural Networks with Continuous Nonlinearities**
N. Frühauf, E. Lüder, Univ Stuttgart, Germany; L.O. Chua, UCB, California, USA 106

11.10 **Counting Stable Equilibria of Cellular Neural Networks - a Graph Theoretic Approach**
P. Kaluzny, Nencki Inst, Warsaw, Poland 112

11.30 **Associative Memory Design Using Space-Varying Cellular Neural Networks**
G. Martinelii , R. Perfetti, Univ. of Roma, Italy 117

11.50 **Theoretical and Experimental Studies of Oscillations in Simple CNN Structures**
M. Ogorzalek, A. Dabrowski, Univ. of Mining and Matallurgy, Krakow, Poland 123

12.10 **Generalized CNN: Potentials of a CNN with Non-Uniform Weights**
M. Balsi, Univ. of Roma, Italy 129

12.30 **Inhomogeneous Cellular Neural Networks: Possibility of Functional Device Design**
Y. Andreyev, Y. Belsky, A. Dmitriev and D. Kuminov, Acad. of Sciences, Moscow, Russia 135

14.20 **Invited Lecture: VLSI-Implementation of CNN**
J.L.Huertas, A. Rodriguez-Vasquez, Univ. of Sevilla, Spain 141

Session 4 VLSI
Chairman: V. Porra, Helsinki Univ., Finland

15.20 **Design of a CMOS Analog Programmable Cellulal Neural Network**
G.F. Dalla Betta, S. Graffi, G. Masetti, Zs. M. Kovacs, Univ. of Bologna 151

15.40 **Multiple-Input OTA Based Circuit for Cellular Neural Network Implementations in VLSI CMOS Technology**
T. Kacprzak, K. Slot, Technical Univ. of Lodz, Poland 157

16.30 **New Test Results of a 4x4 DTCNN Chip**
H. Harrer, J.A. Nossek, TU Munich, Germany 163

16.50 **Design and Testing Issues in Current-Mode Cellular Neural Networks**
S. Espejo, A. Rodriguez-Vasquez and J.L. Huertas, Univ. of Sevilla, Spain 169

17.10 **Optically Realized Feedback Cellular Neural Networks**
K. Slot, Technical Univ. of Lodz, Poland 175

Friday (October 16, 1992)

9.00 **Invited Lecture: Programmability and Applications of the CNN Universal Machine**
T. Roska, Hungarian Academy of Science, Hungary 181

Session 5 Applications
Chairman: W. Mathis, Univ. Wuppertal, Germany

9.40 **Japanese Kanji Character Recognition Using Cellular Neural Networks and Modified Self-Organizing Feature Maps**
K. Nakayama, Y. Chigawa, Kanazawa Univ. Japan 191

10.00 **Nonlinear CNN Cloning Template for Image Thicking**
S. Jankowski, R. Wanczuk, Warsaw Univ. Poland 197

10.20 **An Experimental System for Optical Detection of Layout Errors of Printed Circuit Boards Using Learned CNN Templates**
P. Szolgay, I. Kispal and T. Kozek, Hung. Academy of Science, Hungary 203

11.10 **On the Application of Cellular Automata to Image Thinning with Cellular Neural Network**
D. Yu, Peking Univ, China; Xi. Yu, Beijing Sc. & Techn. Univ.; C. Ho, S. Mori, Keio Univ. Japan 210

11.30 **Optical Tracking System for Automatic Guided Vehicles Using Cellular Neural Networks**
Gy. Eröss, J. Bito, Tungsram Hungary; A. Kiss, J. Vass, Univ. of Veszprem, Hungary; A. Radvanyi, T. Roska, Hung. Acad. of Sciences, Hungary 216

11.50 **A Dual Model of Cyclopean Perception and its Application Potentials in Artificial Stereopsis**
A.G. Radvanyi, Hung. Acad. of Sciences, Hungary 222

12.10 **A Flexible PC-Controlled Analog/Digital Test Set for CNNs**
I. Köhn, B. Smith, H. Harrer, J.A. Nossek, TU Munich, Germany 228

110

Session 6 Theory II
Chairman: U. Ramacher, Siemens AG, Munich, Germany

14.00 **A Geometrical Approach to the Analysis of the Dynamical Behaviour of Neural Networks with Simple Piecewise Linear Limiters**
J. Dehaene, J. Vandewalle, Kath. Univ. Leuven, Belgium 234

14.20 **On the Stability Analysis of Cellular Neural Networks**
F.A. Savaci , Istanbul Univ., Turkey; J. Vandewalle, Kath. Univ. Leuven, Belgium 240

14.40 **Cellular Networks of Oscillators**
A. Barone , M. Balsi, V. Cimagalli , Univ. di Roma, Italy 246

15.00 **Discrete-Time Cellular Neural Networks using Digital Neuron Model with DPLL**
T. Ueda, K. Takahashi, C. Ho, S. Mori, Keio Univ. Hiyoshi, Japan 252

15.50 **Symmetry Properties of Cellular Neural Networks on Square and Hexagonal Grids**
G. Seiler, J. A. Nossek, TU Munich, Germany 258

16.10 **The Reaction-Diffusion Cellular Neural Network**
C.B. Price, P. Wambacq, A. Oosterlinck, Kath Univ. Leuven, Belgium 264

16.30 **Mapping Nonlinear Lattice Equations onto Cellular Neural Networks**
S. Paul, J.A. Nossek, TU Munich, Germany; L.O.Chua, UCB California, USA 270

16.50 **Some novel Capabilities of CNN: Game of Life and multipath algorithms**
L.O. Chua, UCB California, USA; T. Roska, P. Venetianer, A. Zarandy, Hung. Academy of Sciences, Hungary 276

Best Paper Award 282

Authors Index 283

THE CNN UNIVERSAL MACHINE

Part 1: The architecture

Leon.O.CHUA and Tamás ROSKA [+]

The Electronics Research Laboratory and Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley CA 94720 and the [+] Dual and Neural Computing Systems Laboratory in the Computer and Automation Institute of the Hungarian Academy of Sciences, Budapest, Kende-u.13/17, H-1111

Abstract

Various types of CNNs are summarized and the taxonomy of CNN is given according to the different types of grids, processors, interactions, and modes of operation. Next, the CNN Universal Machine is introduced. The architecture and the key features are outlined. A quite exhaustive list of references completes the paper.

1 INTRODUCTION

Since the invention of the Cellular Neural Network [1,2] several extensions [3-8, etc.] formed the CNN (cellular nonlinear network) paradigm. CNN is now an established field, with many scientists, engineers and students contributing to its development. The silicon and optical implementations and experiments promise unusually high computing power.

Our new invention, the CNN Universal Machine and Supercomputer, among others, provide three new capabilities: the analog stored program and the time-multiplex operation of the layers, the wireless detection, and flexible representation of various PDEs using the generalized Chua's circuit for chaos as well.

Next, we will summarize several types of CNNs, as a taxonomy. Section 2 contains the outline of the basic architecture of the CNN Universal Machine. In Section 3 some additional key features are described. A fairly exhaustive, however not complete, list of references is attached. The programmability and key application areas are summarized in a companion paper in this volume.

Various CNN types - a taxonomy

In the next table several types of CNNs are summarized according to grid types, processor types, interaction types, and modes of operation.

Most probably, this table will be expanded continuously. In addition, there are several other issues. For example, the template design and learning, the physical implementations, qualitative theory, accuracy, and the vast area of applications.

CNN:

Cellular anaiog programmable multidimensional processing array

with distributed logic and memory

Grid types	Processor types	Interaction types	Modes of operation
- square, hexagonal tridiagonal	- linear (or small signal operation in the piece-wise-linear sigmoid characteristics)	- linear (one or two variables) memoryless	- continuous-time, discrete-time (synchronous or asynchronous), time-varying
- single and multiple grid-size (coafrse and fine grid, etc)	- sigmoid (including unity gain, high gain, and thresholding)	- nonlinear (one, two, or more variables) memoryless	- local mode and propagating mode
- equidistant and varying grid size (e.g. logarithmic like in the retina)	- Gaussian (inverse Gaussian)	- delay-type	- fully analog or combined with logic (dual computing CNN)
- planar and circular	- first, second and higher order (e.g. one, two, or more capacitors)	- dynamic (lumped)	- fixed template or programmable template (continuously or in discrete values)
- lattice (3D)	- with or without local analog memory	- symmetric and non-symmetric	- transient-, settling-, oscillating-, or chaotic mode
	- with or without local logic		

2 The CNN Universal Machine - the architecture

First, let us define the CNN. CNN is an analog cellular nonlinear dynamic processor array characterized by the following features:

(i) the analog processors are processing continuous signals, they are continuous-time or discrete-time signals (i.e. the input, output, and state variables are C^k functions at least for finite intervals), the processors are basically identical (a single term of them may vary regularly);

(ii) the processors are placed on a 3D geometric cellular grid (several 2D layers);

(iii) the interactions between the processors are mainly local, i.e. each processor is interacting with others within a finite neighborhood (nearest neighbor is just a special case), and mainly translation invariant defined by the cloning template (the dynamics of a given interaction may be a design parameter);

(iv) the mode of operation may be: transient, equilibrium, periodic, chaotic, or combined with logic (without A-D conversion!).

In the CNN Universal Machine such a CNN array is embedded in an analogic (analog and logic) stored program architecture. This architecture is shown in Figure 1.

In addition to the CNN nucleus, each cell contains several local analog memory cells (LAM) and a single local logic memory (LLM). LAM cell values are combined by a local analog output unit (LAOU) while the logic register values are combined by a local logic unit (LLU). The local communication and control unit (LCCU) provides the control of the cell configuration and the analog and logic instruction communication. Some parts of this cell have already been implemented in various CNN circuits.

The entire cell array is controlled by a global analogic programming unit (GAPU) containing an analog program register (APR), a logic program register (LPR), a switch configuration register (SCR), and a global analogic control unit (GACU). This unit is a key for implementing the analog stored program. If the Universal CNN Machine is implemented in a single chip, the CNN universal chip, several of them can be used in an array partitioning the GAPU between the chips and a central GAPU. Then, the APR could be divided, to form an analog cache register.

It can be shown that the local regular connectivity is essential for implementing a stored program analogic algorithm on silicon.

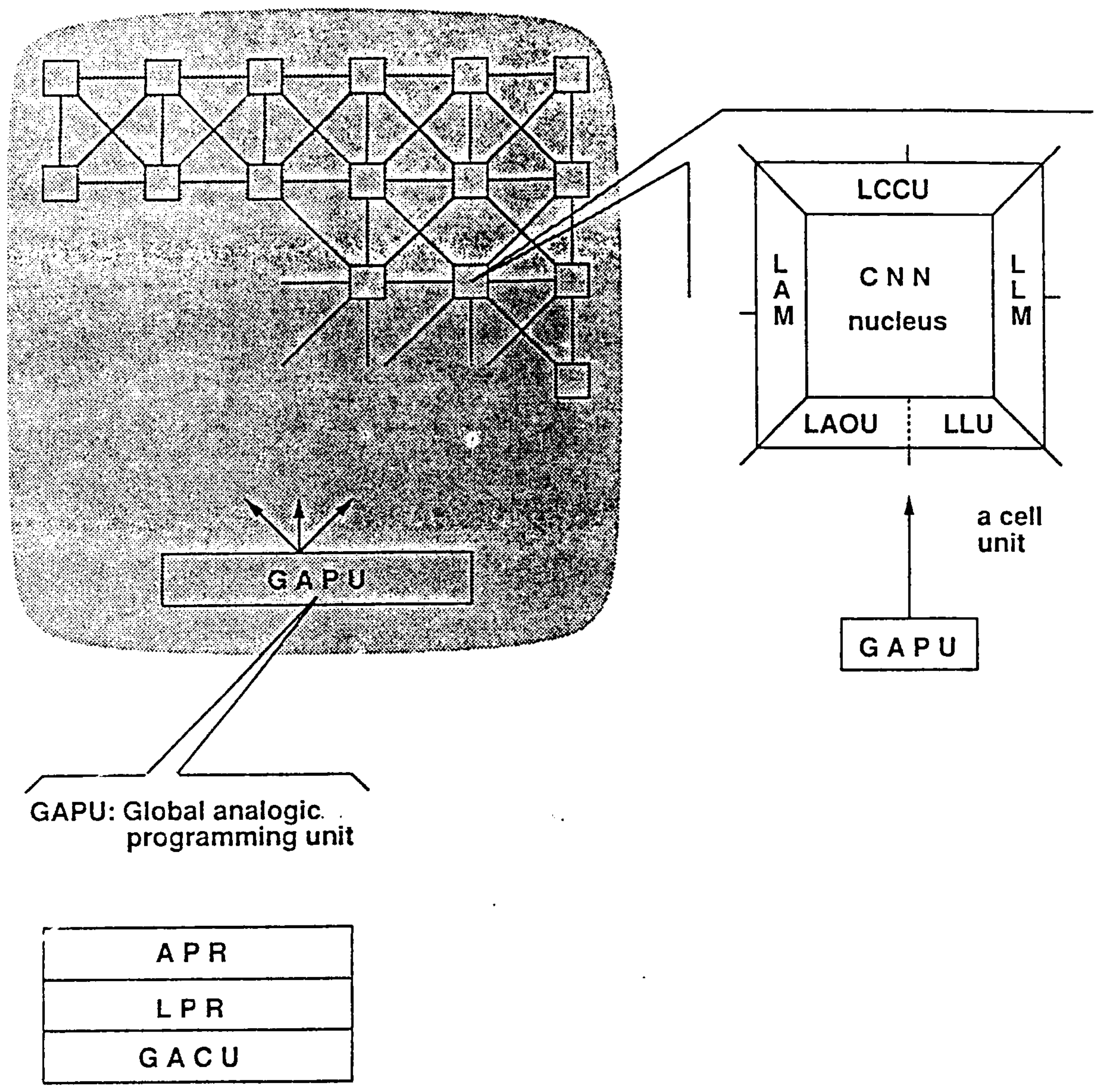


Figure 1

3 SOME OTHER FEATURES

Recently, we have found templates [78] which detect some features by switching the pertinent cells into an oscillation regime. This allows the detection by a wireless way, thereby drastically increasing the output throughput.

Using various simple dynamic circuits as processors, several types of partial differential equations (PDEs) can be modelled in a discretized space. The diffusion equation [1,2] and the wave equation [7, 64, 70] are such examples. We can use the generalized Chua's circuit of Figure 2 as a processor (the nonlinear resistor is the so called Chua's diode). Even if the interactions are represented by a simple resistive grid, and just a part of the circuit is used, unique phenomena can be generated [61].

As to the silicon implementation of the inductor, several equivalent circuits can be used. One possibility is the use of a gyrator which can simply be implemented by using two voltage controlled current sources (or transconductances, used extensively in the CNN and general neural network implementations).

The CNN Universal Machine's analogic programs define a new world of analogic (dual) type of software. All these functionalities can be implemented by using the CNN Workstation [79]. The special language for defining these algorithms contains some analog instructions as well.

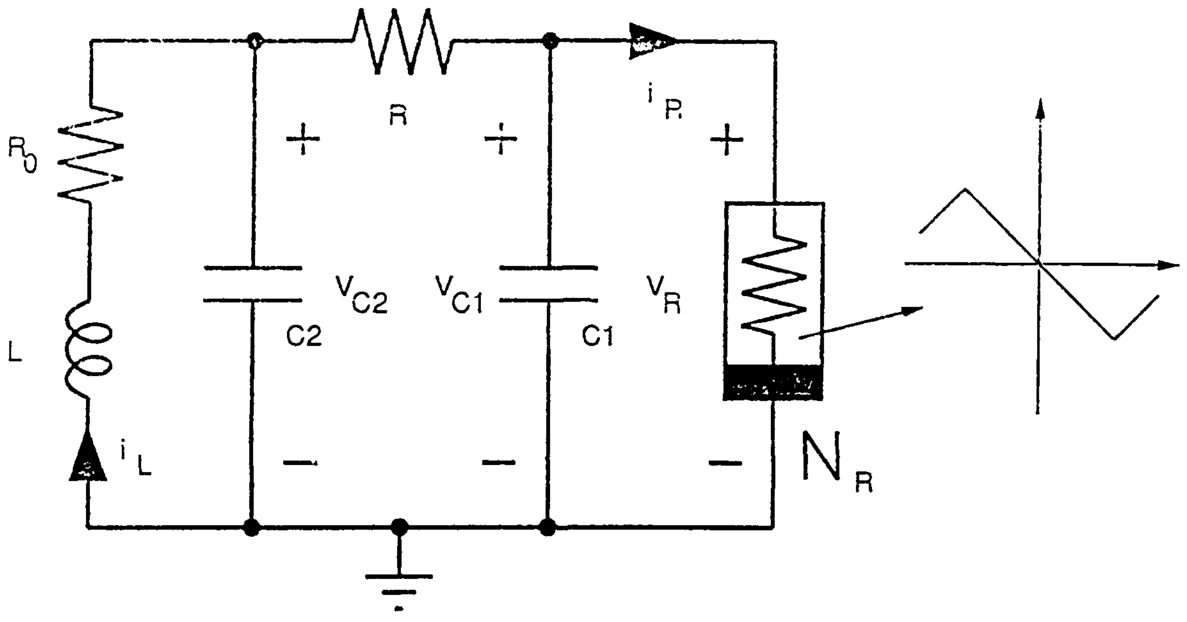


Figure 2

Beside the local connections, without increasing the connection density, some global wires can also be used in the CNN Universal Machine. Local and global dynamics are interacting in this way.

ACKNOWLEDGEMENTS

This work has been supported by the Office of Naval Research under Grant N00014-89-J-1402, by the National Science Foundation under grant MIP 86-14000, and by the National Science Foundation under Grant INT 90-01336 in cooperation with the Hungarian Academy of Sciences.

References

[1] L. O. Chua and L. Yang. Cellular Neural Networks: Theory. *IEEE Transactions on Circuits and Systems*, 35:1257–1272, October 1988.

[2] L. O. Chua and L. Yang. Cellular Neural Networks: Applications. *IEEE Transactions on Circuits and Systems*, 35:1273–1290, October 1988.

[3] T. Roska and L. O. Chua. Cellular Neural Networks with nonlinear and delay-type template elements. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 12–25, 1990. extended version in Int. J. Circuit Theory and Applications, Vol.20, 1992.

[4] L. O. Chua, T. Roska, P. L. Venetianer, and A. Zarandy. Some novel capabilities of CNN: Game of life and examples of mulitpath algorithms. Report DNS-3-1992, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1992.

[5] J. A. Nossek, G. Seiler, T. Roska, and L. O. Chua. Cellular Neural Networks: Theory and circuit design. Report TUM-LNS-TR-90-7, Technische Universität München, December 1990. to appear in Int. J. Circuit Theory, Vol.20, 1992.

[6] H. Harrer and J. A. Nossek. Time discrete Cellular Neural Networks: Architecture, applications and realization. Report TUM-LNS-TR-90-12, Technische Universität München, November 1990.

[7] J. Henseler and P. J. Braspennig. Memb-ain: a cellular neural network model based on a vibrating membrane. to appear in Int. J. Circuit Theory and Applications, Vol.20,, 1992.

[8] A. Radvanyi, K. Halonen, and T. Roska. The CNNL simulator and some time-varying CNN templates. Report DNS-9-1991, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1992.

[9] L. O. Chua and B. E. Shi. Exploiting Cellular automata in the design of Cellular Neural Networks for binary image processing. Memorandum UCB/ERL M89/130, University of California at Berkeley Electronics Research Laboratory, November 1989.

[10] L. O. Chua and T. Roska. Stability of a class of nonreciprocal cellular neural networks. *IEEE Transactions on Circuits and Systems*, 37:1520–1527, December 1990.

[11] V. Cimagalli. A neural network architecture for detecting moving objects II. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 124–126, 1990.

[12] N. Frühauf and E. Lüder. Realizations of CNNs by optical parallel processing with spatial light valves. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 281–291, 1990.

[13] K. Halonen, V. Porra, T. Roska, and L. Chua. VLSI implementation of a reconfigurable Cellular Neural Network containing local logic (CNNL). In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 206–215, 1990. extended version in Int. J. Circuit Theory and Applications, Vol.20, 1992.

[14] T. Matsumoto, L. O. Chua, and R. Furukawa. CNN cloning template: Hole-filler. *IEEE Transactions on Circuits and Systems*, 37:635–638, May 1990.

[15] T. Matsumoto, L. O. Chua, and H. Suzuki. CNN cloning template: Shadow detector. *IEEE Transactions on Circuits and Systems*, 37(8):1070 – 1073, August 1990.

[16] T. Matsumoto, L. O. Chua, and H. Suzuki. CNN cloning template: Connected component detector. *IEEE Transactions on Circuits and Systems*, 37:633-635, May 1990.

[17] T. Matsumoto, T. Yokohama, H. Suzuki, and R. Furukawa. Several image processing examples by CNN. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 100–111, 1990.

[18] T. Matsumoto, L. O. Chua, and T. Yokohama. Image thinning with a Cellular Neural Network. *IEEE Transactions on Circuits and Systems*, 37:633–635, May 1990.

[19] J. A. Nossek and G. Seiler. An equivalence between multi-layer perceptrons with step function type nonlinearity and a class of Cellular Neural Networks. Report TUM-LNS-TR-90-7, Technische Universität München, December 1990.

[20] T. Roska, T. Boros, P. Thiran, and L. O. Chua. Detecting simple motion using Cellular Neural Networks. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 127–138, 1990.

[21] T. Roska, G. Bartfai, P. Szolgay, T. Sziranyi, A. Radvanyi, T. Kozek, Zs. Ugray, and A. Zarandy. A digital multiprocessor hardware accelerator board for cellular neural networks: CNN-HAC. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 160–168, 1990. extended version in Int. J. Circuit Theory and Applications, Vol.20, 1992.

[22] T. Roska and A. Radvanyi. *CNND simulator, Cellular Neural Network embedded in a simple Dual computing structure, User's guide Version 3.0.* Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1990. Report No. 37/1990.

[23] G. Seiler. Small object counting with Cellular Neural Networks. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 114–123, 1990.

[24] K. Ślot. Determination of Cellular Neural Network parameters for feature detection of two dimensional images. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 82–91, 1990.

[25] L. Vandenberghe and J. Vandewalle. Finding multiple equilibrium points of Cellular Neural Networks without enumeration. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 45–55, 1990. to appear in Int. J. of Circuit Theory and Applications, Vol.20, 1992.

[26] J. E. Varrientos, J. Ramirez-Angulo, and E. Sanchez-Sinencio. Cellular neural networks implementation: a current-mode approach. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 216–225, 1990.

[27] L. Yang, L. O. Chua, and K. R. Krieg. VLSI implementation of Cellular Neural Networks. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 2425–2427. IEEE, 1990.

[28] G. G. Yang. Optical associative memory with invariance. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 291–302, 1990.

[29] F. Zou, S. Schwarz, and J. A. Nossek. Cellular neural network design using a learning algorithm. In *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pages 73–81, 1990.

[30] F. Zou and J. A. Nossek. Stability of Cellular Neural Networks with opposite sign templates. Report TUM-LNS-TR-15, Technische Universität München, December 1990.

[31] M. Balsi. Remarks on the stability and functionality of CNN's with one-dimensional templates. Report DNS-5-1991, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1991. under publication in Int. J. Circuit Theory and Applications.

[32] T. Boros, K. Lotz, A. Radvanyi, and T. Roska. Some useful, new, noninear and delay-type templates. Report DNS-1-1991, Dual and Neural Computing Systems Res. Lab., Comp. Aut. Inst., Hung. Acad. Sci. (MTA SzTAKI), Budapest, 1991.

[33] L. O. Chua and B. E. Shi. Multiple layer Cellular Neural Networks: A tutorial. In E. F. Deprettere and A. van der Veen, editors, *Algorithms and Parallel VLSI Architectures*, volume A: Tutorials, pages 137–168. Elsevier Science Publishers B. V., New York, 1991.

[34] L. O. Chua and P. Thiran. An analytic method for designing simple cellular neural networks. *IEEE Transactions on Circuits and Systems*, 38:1332–1341, 1991.

[35] L. O. Chua, L. Yang, and K. R. Krieg. Signal processing using cellular Neural Networks. *Journal of VLSI Signal Processing*, 3:25–52, 1991.

[36] P. P. Civalleri, M. Gilli, and L. Pandolfi. On stability of cellular neural networks with delay. Report, Politecnico di Torino, November 1991. under publication.

[37] K. R. Crounse, T. Roska, and L. O. Chua. Image halftoning with Cellular Neural Networks. Memorandum UCB/ERL M91/106, University of California at Berkeley Electronics Research Laboratory, November 1991. submitted for publication.

[38] J. M. Cruz and L. O. Chua. A CNN chip for connected component detection. *IEEE Transactions on Circuits and Systems*, 38(7):812–817, July 1991.

[39] N. Frühauf, E. Lüder, M. Gaiada, and G. Bader. An optical implementation of space invariant Cellular Neural Networks. In *European Conference on Circuit Theory and Design*, pages 42–51, 1991.

[40] S. Fukuda, T. Boros, and T. Roska. A new efficient analysis of thermographic images by using cellular neural networks. Technical Report DNS-11-1991, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1990. under publication.

[41] C. Guzelis and L. O. Chua. Stability analysis of generalized cellular neural networks. Technical Report UCB/ERL M91/23, University of California at Berkeley Electronics Research Laboratory, March 1991. under publication.

[42] H. Harrer and J. A. Nossek. An analog CMOS compatible convolution circuit for analog neural networks. Report TUM-LNS-TR-91-11, Technische Universität München, 1991. Extended version in IEEE Tr. Neural Networks 1992.

[43] H. Harrer, J. A. Nossek, and R. Stelzl. An analog implementation of discrete-time cellular neural networks. Report TUM-LNS-TR-91-14, Technische Universität München, June 1991.

[44] V. I. Krinsky, V. N. Biktashev, and I. R. Efimov. Autowave principles for parallel image processing. *Physica D*, 49:247–253, 1991.

[45] A. Radvanyi and T. Roska. The CNN workstation - CNND version 4.1. Technical Report DNS-12-1991, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1991. submitted for publication.

[46] T. Roska, A. Radvanyi, T. Kozek, and T. Boros. Dual CNN software library. Report DNS-7-1991, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1991.

[47] T. Roska. Cellular neural networks: a state-of-the-art review. In *Proc. European Conference on Circuit Theory and Design (ECCTD-91)*, pages 1–9, 1991.

[48] T. Roska and P. Szolgay. A comparison of various cellular neural network (CNN) realizations - a review. In *Proc. 2nd Int. Conf. on Microelectronics for Neural Networks*, pages 423–421, 1991.

[49] T. Roska. Dual computing structures containing analog cellular neural networks and digital decision units. In *Proc. IFIP Workshop on Silicon Architectures for Neural Nets*, pages 233–244, Nice, 1990, 1991. North Holland, Amsterdam.

[50] S. Schwarz and W. Mathis. A design algorithm for Cellular Neural Networks. In *Proceedings of the 2nd International Conference on Microelectronics for Neural Networks*, pages 53–59, 1991.

[51] G. Seiler and M. Hasler. Convergence of reciprocal cellular neural networks. Report TUM-LNS-TR-91-12, Technische Universität München, June 1991.

[52] B. E. Shi and L. O. Chua. A generalized Cellular Neural Network of a novel edge detection algorithm. Memorandum UCB/ERL M91/25, University of California at Berkeley Electronics Research Laboratory, April 1991.

[53] P. Szolgay and T. Kozek. Optical detection of layout errors of printed circuit boards using learned CNN templates. Report DNS-8-1991, Dual and Neural Computing Systems Res. Lab., Comp. Aut. Inst., Hung. Acad. Sci. (MTA SzTAKI), Budapest, 1991.

[54] T. Roska, C. W. Wu, M. Balsi, and L. O. Chua. Stability and dynamics of delay-type and nonlinear Cellular Neural Networks. Memorandum UCB/ERL M91/110, University of California at Berkeley Electronics Research Laboratory, December 1991.

[55] L. O. Chua and C. W. Wu. On the universe of stable cellular neural networks. ERL Memorandum UCB/ERL M91/31, University of California, Berkeley, 1991. to appear in Int. J. of Circuit Theory and Applications, Vol.20, 1992.

[56] F. Zou and J. A. Nossek. A chaotic attractor with Cellular Neural Networks. *IEEE Transactions on Circuits and Systems*, 38(7):811–812, July 1991.

[57] J. M. Cruz and L. O. Chua. Design of high speed high density CNNs in CMOS technology. *International Journal of Circuit Theory and Applications*, 20(4), 1992.

[58] Gy. Eröss, A. Radvanyi, T. Boros, Á. Kiss, P. Lantos, J. Bitó, T. Roska, and J. Vass. Optical tracking system for automatic guided vehichle (AGV) using cellular neural networks. Report DNS-15-1992, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1992. submitted for publication.

[59] W. Heiligenberg and T. Roska. On biological sensory information processing principles relevant to dually computing CNNs. Report DNS-4-1992, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1992.

[60] T. Kozek, T. Roska, and L. O. Chua. Genetic algorithm for CNN template learning. Memorandum, University of California at Berkeley Electronics Research Laboratory, 1992. submitted for publication.

[61] V. Pèrez-Muñuzuri, V. Pèrez-Villar, and L. O. Chua. Autowaves for image processing on a two-dimensional CNN array of Chua's circuits: flat and wrinkled labyrinths. Memorandum, University of California at Berkeley Electronics Research Laboratory, 1992. submitted for publication.

[62] T. Roska, J. Hámori, E. Lábos, K. Lotz, L. Orzó, J. Takács, P. Venetianer, Z. Vidnyánszky, and Á. Zarándy. The use of CNN models in the subcortical visual pathway. Technical report, submitted for publication, 1992.

[63] T. Roska and L. O. Chua. The dual CNN analog software. Technical Report DNS-2-1992, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1992.

[64] T. Roska. Programmable cellular neural networks - a state-of-the-art. In P. Dewilde and J. Vandewalle, editors, *State-of-the-art in computer systems and software engineering*. Kluwer Academic Publishers, 1992.

[65] T. Roska, C. W. Wu, M. Balsi, and L. O. Chua. Stability and dynamics of delay-type general and Cellular Neural Networks. *IEEE Transactions on Circuits and Systems*, 39(6):487–490, June 1992.

[66] T. Roska, T. Boros, A. Radvanyi, P. Thiran, and L. O. Chua. Detecting moving and standing objects using cellular neural networks. *Int. J. Circuit Theory and Applications*, July-August 1992.

[67] T. Roska and L. O. Chua. CNN: cellular analog programmable multidimensional processing array with distibuted logic and memory. submitted for publication, June 1992.

[68] B. E. Shi, T. Roska, and L. O. Chua. Design of linear cellular neural networks for motion sensitive filtering. submitted for publication, 1992.

[69] P. Szolgay, I. Kispál, and T. Kozek. An experimental system for optical detection of layout errors using analog software on a dual CNN. Report DNS-14-1992, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1992. under publication.

[70] P. Szolgay, G. Vörös, and Gy. Erőss. On the application of cellular neural network (CNN) paradigm in mechanical vibrating systems. Report DNS-8-1992, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1992. submitted for publication.

[71] L. Vandenberghe, J. Vandewalle, St. Daenen, and M. Verduyn. Measuring the size of an object by a sequence of CNNs. Memorandum, Catholic University, Leuven, 1992.

[72] *Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA-90)*, 1990. Budapest, IEEE Cat. No. 90TH0312-9.

[73] *Proceedings of the 2nd IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA-92)*, 1992. Munich.

[74] European Conference on Circuit Theory and Design (ECCTD-91). *Proceedings of the Special Session on Cellular Neural Networks*, September 1991.

[75] Int. J. Circuit Theory and Applications. *Special issue on Cellular Neural Networks*, July-August 1992.

[76] IEEE Transactions on Circuits and Systems. *Special issue on Cellular Neural Networks*, March 1993.

[77] T. Roska. Analog events and a dual computing structure using analog and digital circuits and operators. In *Discrete Event Systems: Models and Applications*, pages 225–238, 1988. eds. P.Varaiya and A.B.Kurzhanski, Springer Verlag.

[78] C.W. Wu, T. Roska, and L. O. Chua. Detecting features by CNNs in oscillating or chaotic mode. in preparation, 1992.

[79] CNN workstation (version 5.1). In *MTA SzTAKI papers, Users' documentation*, 1992. Budapest.

Exact Design of Reciprocal Cellular Neural Networks

J.A. Osuna and G.S. Moschytz

Swiss Federal Institute of Technology Zurich
Signal and Information Processing Laboratory
Sternwartstrasse 7, CH-8092 Zürich, Switzerland
Tel. +41 1 256 3620; Fax +41 1 262 0823
Email osuna@isi.ethz.ch

- *Abstract.* **Based on two classes of equilibrium equations, a design method for reciprocal Cellular Neural Networks is presented. The local rules defining the task to be accomplished by the network are directly mapped into a set of linear inequalities that bound the solution space of the network parameters for the given problem. All points in the solution space guarantee the correct operation of the network. A solution can be computed by the relaxation method for solving sets of linear inequalities.**

1 Introduction

A Cellular Neural Network (CNN) performs a nonlinear mapping of an input onto an output [1, 2]. The mapping is completely defined by the space-invariant network parameters. Each application such as, e.g., edge extraction or shadowing needs its own parameter values to process the input to the desired output. Locally connected CNNs with space-invariant parameters are able to process tasks which can completely be described by local rules involving only neighbouring cells.

Several design methods for synthesizing CNNs have been proposed. In [3] a training rule was presented to determine the weights of the network from input image patterns and desired output image patterns. By learning from examples, it is assumed that, after training, the network correctly processes inputs which have never been shown before. Unfortunately, however, the proposed design method does not even guarantee the desired outputs for a given input learning set.

An analytic method for designing simple CNNs has also been published [4]. It uses rules that explicitly describe the task to be accomplished by the network. These rules establish a set of inequalities that must be satisfied by the network parameters. A solution of this set of inequalities guarantees correct operation of the network. Nevertheless, it may be difficult to construct such a set of inequalities for a given task, and for some cases the method is too restrictive, resulting in an empty solution space.

Our design method maps the rules that define the application directly into a set of linear inequalities using two classes of equilibrium equations: The first class contains *desired* equilibrium states, the second contains *forbidden* ones. Any point in the solution space bounded by the set of linear inequalities guarantees correct operation of the network for a given task. The parameter values can be computed by the relaxation method for the solution of sets of linear inequalities [3].

2 Desired and Forbidden Equilibrium States

The first-order nonlinear differential equations defining the dynamics of a reciprocal CNN with neighbourhood $N_1(i,j)$ (radius equal to 1), and with binary input values can be written as follows:

$$\frac{dx_{ij}}{dt} + x_{ij} = \boldsymbol{y}_{ij}^T\boldsymbol{a} + \boldsymbol{u}_{ij}^T\boldsymbol{b} + c \quad \text{(1a)}$$
$$= \boldsymbol{\gamma}_{ij}^T\boldsymbol{x}$$
$$1 \le i \le M\ ,\ 1 \le j \le N$$

where

$$\boldsymbol{\gamma}_{ij}^T = \left[\boldsymbol{y}_{ij}^T\ \boldsymbol{u}_{ij}^T\ 1\right] \quad \text{(1b)}$$
$$= \left[\gamma_1^{ij}\ \gamma_2^{ij}\ \cdots\ \gamma_{19}^{ij}\right]$$
$$\boldsymbol{x}^T = \left[\boldsymbol{a}^T\ \boldsymbol{b}^T\ c\right] \quad \text{(1c)}$$
$$= [\ a_1\ \ a_2\ \cdots\ a_9\ \ b_1\ \ b_2\ \cdots\ b_9\ \ c\ ]$$
$$y_{ij}(t) = \frac{1}{2}(|x_{ij}(t)+1| - |x_{ij}(t)-1|) \quad \text{(1d)}$$

(piecewise-linear function)

The CNN is subjected to the following restrictions:

$$|u_{ij}| = 1\ , \quad \forall\, i,j \quad \text{(1e)}$$

(binary input)

$$\begin{array}{l} a_1 = a_9\ ,\ a_2 = a_8 \\ a_3 = a_7\ ,\ a_4 = a_6 \end{array} \quad \text{(1f)}$$

(symmetry condition)

$$a_5 > 1 \quad \text{(1g)}$$

(parameter assumption)

Table 1 illustrates the correspondence between vector $\boldsymbol{a}$ and the space-invariant cloning template A.

a_1	a_2	a_3
a_4	a_5	a_6
a_7	a_8	a_9

Table 1: Component correspondence for vector $\boldsymbol{a}$ and template A

The output values y_{ij} of the MxN CNN converge to ±1 for $t \to \infty$ because of symmetry condition (1f) (reciprocal CNN), parameter assumption (1g) and the sigmoidal characteristic (1d) of the nonlinear function [1]. This means, that each vector component γ_r^{ij}, $1 \le r \le 9$, converges to ±1 for $t \to \infty$. On the other hand, because of the binary-input condition (1e), $|\gamma_r^{ij}| = 1$, $10 \le r \le 18$, for $t \ge 0$. Thus, $\boldsymbol{\gamma}_{ij}(t \to \infty) = \boldsymbol{\gamma}_{ij}^{\infty} \in \{-1,\ 1\}^{19}$ contains binary components ±1 corresponding to output and input values in the neighbourhood $N_1(i,j)$ of cell $C(i,j)$. [1]

Let Γ be the set of all possible vectors $\boldsymbol{\gamma}_{ij}^{\infty}$ with different component values. Next let Γ_d and Γ_f be disjoint subsets of Γ containing those desired and forbidden vectors ${}^d\boldsymbol{\gamma}_{ij}^{\infty}$ and ${}^f\boldsymbol{\gamma}_{ij}^{\infty}$, whose components correspond to input/output combinations which are prescribed by, resp. should not appear in, a specific application, i.e. a specific mapping function F

$$F:\ \{-1,\ 1\}^{M\times N} \longrightarrow \{-1,\ 1\}^{M\times N} \quad (2)$$

Then for all $\boldsymbol{\gamma}_{ij}^{\infty} \in \Gamma_d$, i.e. for vectors ${}^d\boldsymbol{\gamma}_{ij}^{\infty}$, the following inequalities formulate the desired conditions that the parameter vector $\boldsymbol{x}$ must satisfy in order to enable the CNN to settle in a desired output state:

$$\boldsymbol{\gamma}_{ij}^T\boldsymbol{x} \ge\ \ 1, \quad \text{for } \gamma_5^{ij} =\ \ 1 \quad \text{(3a)}$$
$$\boldsymbol{\gamma}_{ij}^T\boldsymbol{x} \le -1, \quad \text{for } \gamma_5^{ij} = -1 \quad \text{(3b)}$$

For clarity, the shorter notation $\boldsymbol{\gamma}_{ij}^T$ has been used instead of $({}^d\boldsymbol{\gamma}_{ij}^{\infty})^T$. [2]

On the other hand, vectors ${}^f\boldsymbol{\gamma}_{ij}^{\infty}$ belonging to the set Γ_f of forbidden combinations

[1]The last component of vector $\boldsymbol{\gamma}_{ij}^{\infty}$ is always equal to unity due to $\gamma_{19}^{ij} := 1$ (see (1b)), and is not related to input or output values.

[2]Inequalities (3) result from equations (1a) and (1d), and from the facts that $\frac{dx_{ij}}{dt} = 0$ and $y_{ij} = \pm1$ for $t \to \infty$ [1].

define the following inequalities that prevent the transient of the CNN to converge to an undesired output state for a given input:

$$\gamma_{ij}^T \boldsymbol{x} < \ \ 1, \quad \text{for } \gamma_5^{ij} = \ \ 1 \qquad (4a)$$

$$\gamma_{ij}^T \boldsymbol{x} > -1, \quad \text{for } \gamma_5^{ij} = -1 \qquad (4b)$$

Here γ_{ij}^T stands for $({}^f\gamma_{ij}^\infty)^T$. With inequalities (4) and for a fixed parameter vector $\boldsymbol{x}$, any state variable x_{ij} that produces an undesired output $y_{ij} = \pm 1$ is pushed back from the outer parts of the piecewise-linear function into the linear region $|x_{ij}| < 1$ in order to let the CNN search for another equilibrium state. If the resulting vector γ_{ij} of this new equilibrium state also belongs to Γ_f then the state variable is pushed back again until a desired equilibrium state is reached. Inequalities (3) assure that such an equilibrium state exists.

In summary, inequalities (3) and (4) together with (1g) and symmetry condition (1f) bound the solution space of parameter vector $\boldsymbol{x}$. The correct operation of the CNN for any point within the solution space is guaranteed by this procedure. The elements of Γ_d and Γ_f can directly be derived from the local rules defining F, as demonstrated by the next example.

3 Example: Edge Extraction

The task of extracting the edge of a figure can be completely described by local rules which only involve cells $C(k,l)$ belonging to the corresponding neighbourhood $N_1(i,j)$. This is essential because applications which need the direct influence of a cell $C(k,l) \notin N_1(i,j)$ on $C(i,j)$ cannot be realized with the common definition (1) of a CNN. The advantage of using local rules for the description of the task is that the correct operation of the network does not depend on its size, i.e. on the values of constants M and N. The rules for edge extraction can be stated as follows (compare with Figure 1):

A cell does not belong to the edge of a figure if

- the cell does not belong to the figure itself, i.e. if its input value is white (or, equivalently, -1)
- the cell is in the inner part of the figure, i.e. if its input value is black (or equivalently 1) *and* the input values of cells $C(i-1,j)$, $C(i,j-1)$, $C(i,j+1)$ and $C(i+1,j)$ are also black.

These rules deal only with five cells, namely the center cell and the off-diagonal cells within the neighbourhood. For the center cell the input and the output values are required, for the off-diagonal cells only the input value is needed (see Table 2). Moreover,

	u_2^{ij}	
u_4^{ij}	u_5^{ij}	u_6^{ij}
	u_8^{ij}	

y_5^{ij}

Table 2: inputs and the output involved in the edge-extraction problem

the space symmetry of the edge-extraction problem allows us to weight the off-diagonal input values with the same parameter $b_\#$. Thus, a reduced parameter vector is sufficient for the correct operation of the network:

$$\boldsymbol{x}^T = [\, a_5 \ \ b_\# \ \ b_5 \ \ c \,], \qquad (5a)$$

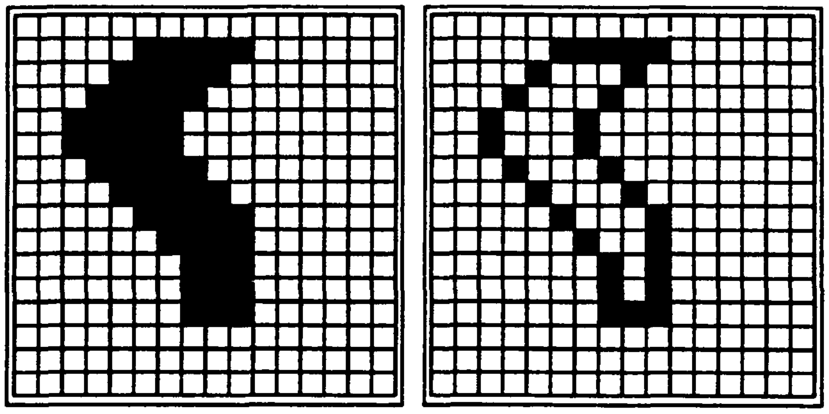
where

$$b_\# = b_2 = b_4 = b_6 = b_8$$

Vectors γ_{ij} and γ_{ij}^∞ are modified and now contain only the needed input and output values having again the same size as parameter vector $\boldsymbol{x}$:

$$\gamma_{ij}^T = \left[\gamma_5^{ij} \ \ \gamma_{11}^{ij} + \gamma_{13}^{ij} + \gamma_{15}^{ij} + \gamma_{17}^{ij} \ \ \gamma_{14}^{ij} \ \ 1\right]$$

$$\Rightarrow \gamma_{ij}^\infty = \left[{}^\bullet\gamma_1^{ij} \ \ {}^\bullet\gamma_2^{ij} \ \ {}^\bullet\gamma_3^{ij} \ \ 1\,\right]^T \qquad (5b)$$



(a) (b)

Figure 1: (a) input pattern and
(b) the desired output pattern for edge extraction in a 16x16 layer [3]

Vector component ${}^*\gamma_1^{ij}$ corresponds to the output of cell $C(i,j)$, ${}^*\gamma_3^{ij}$ contains its input value, and ${}^*\gamma_2^{ij}$ is the sum of the off-diagonal inputs around cell $C(i,j)$. ${}^*\gamma_3^{ij}$ and ${}^*\gamma_1^{ij}$ can take on values ± 1, i.e. the cell's input and output are either black or white.

Next, we have to differentiate between cells at a corner, at the border, or in the inner part of the CNN in order to compute the values of vector component ${}^*\gamma_2^{ij}$. It is obvious that for an inner cell ${}^*\gamma_2^{ij}$ can take on values $-4, -2, 0, 2, 4$ depending on the sum of white ($= -1$) and black ($= 1$) off-diagonal cells[4] around cell $C(i,j)$. For example, if $C(i,j)$ is surrounded by white off-diagonal cells then ${}^*\gamma_2^{ij} = -4$, but if one of these cells is black then ${}^*\gamma_2^{ij} = -3+1 = -2$. For border cells, one of the elements of the sum $\gamma_{11}^{ij}+\gamma_{13}^{ij}+\gamma_{15}^{ij}+\gamma_{17}^{ij}$ is zero because there cell $C(i,j)$ is surrounded only by three off-diagonal cells. For corner cells even two elements of this sum disappear. The following relations summarize the results:

$$ {}^*\gamma_1^{ij},\ {}^*\gamma_3^{ij} \in \{-1,\ 1\} \qquad (6a)$$

$$\begin{aligned} {}^*\gamma_2^{ij} &\in \{-2,\ 0,\ 2\} \\ &\quad \text{(for corner cells)} \\ {}^*\gamma_2^{ij} &\in \{-3, -1,\ 1,\ 3\} \\ &\quad \text{(for border cells)} \\ {}^*\gamma_2^{ij} &\in \{-4, -2,\ 0,\ 2,\ 4\} \\ &\quad \text{(for center cells)} \end{aligned} \qquad (6b)$$

From the first rule of the edge-extraction problem it follows that if the input of a cell is white then the cell has to produce a white output, i.e. whenever ${}^*\gamma_3^{ij} = -1$ the output ${}^*\gamma_1^{ij}$ must also be -1. The second rule states that whenever ${}^*\gamma_3^{ij} = 1$ (black input), ${}^*\gamma_1^{ij}$ also has to be 1 unless the inputs of the off-diagonal cells are all black, i.e. ${}^*\gamma_2^{ij} = 4$. From this, the set Γ_d can be constructed in a straight-forward procedure assigning to it all vectors γ_{ij}^{∞} whose components accomplish one of the following conditions:

$(i)\ {}^*\gamma_3^{ij} = -1 \ \wedge\ {}^*\gamma_1^{ij} = -1$

$(ii)\ {}^*\gamma_3^{ij} = 1 \ \wedge\ {}^*\gamma_1^{ij} = 1 \ \wedge$
$\quad {}^*\gamma_2^{ij} \in \{-4, -3, -2, -1,\ 0,\ 1,\ 2,\ 3\}$

$(iii)\ {}^*\gamma_3^{ij} = 1 \ \wedge\ {}^*\gamma_1^{ij} = -1 \ \wedge\ {}^*\gamma_2^{ij} = 4$

Equation (7) shows the result.

[3]We take the same two-dimensional image as was used in [3] to train the network.

[4]The input values of the cells are added.

$$\Gamma_d = \left\{ \begin{bmatrix}-1\\-2\\-1\\1\end{bmatrix}, \begin{bmatrix}-1\\0\\-1\\1\end{bmatrix}, \begin{bmatrix}-1\\2\\-1\\1\end{bmatrix}, \begin{bmatrix}1\\-2\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\2\\1\\1\end{bmatrix}, \right. \quad \text{corner cells \& inner cells}$$

$$\begin{bmatrix}-1\\-3\\-1\\1\end{bmatrix}, \begin{bmatrix}-1\\-1\\-1\\1\end{bmatrix}, \begin{bmatrix}-1\\1\\-1\\1\end{bmatrix}, \begin{bmatrix}-1\\3\\-1\\1\end{bmatrix}, \begin{bmatrix}1\\-3\\1\\1\end{bmatrix}, \begin{bmatrix}1\\-1\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\1\\1\end{bmatrix}, \begin{bmatrix}1\\3\\1\\1\end{bmatrix}, \quad \text{border cells} \qquad (7)$$

$$\left. \begin{bmatrix}-1\\-4\\-1\\1\end{bmatrix}, \begin{bmatrix}-1\\4\\-1\\1\end{bmatrix}, \begin{bmatrix}1\\-4\\1\\1\end{bmatrix}, \begin{bmatrix}-1\\4\\1\\1\end{bmatrix} \right\} \quad \text{inner cells}$$

Γ_f can easily be constructed using the disjoint property of both sets Γ_d and Γ_f and the fact that *all* other vectors γ_{ij}^{∞} whose components do not accomplish any of the conditions (*i*) – (*iii*) lead to forbidden equilibrium states:

$$\Gamma_f = \Gamma \backslash \Gamma_d \qquad (8)$$

We now have all the inequalities that bound the solution space of parameter vector $\boldsymbol{x} \in \mathbb{R}^4$ such that any point in the solution space guarantees the correct operation of the CNN for edge extraction, independently of the sizes M and N and the initial state[5]. A solution that satisfies inequalities (1g), (3) and (4) can be computed by the relaxation method for solving sets of linear inequalities [3]. The following equation gives a solution of the parameter vector $\boldsymbol{x}$ for the edge-extraction problem:

$$\boldsymbol{x} = \begin{bmatrix} 1.0559 \\ -0.2615 \\ 1.2436 \\ -0.3419 \end{bmatrix} \qquad (9)$$

Table 3 orders the same result in template notation.

[5]The local rules for edge extraction do not involve the initial state of the CNN.

0	−0.2615	0
−0.2615	1.2436	−0.2615
0	−0.2615	0

cloning template B

0	0	0
0	1.0559	0
0	0	0

cloning template A

$I = -0.3419$

constant current source

Table 3: Edge Extraction: same parameter set as in equation (9) ordered in template notation

4 Conclusions

An exact design method for reciprocal Cellular Neural Networks with binary inputs has been presented. Given an application, two disjoint sets of vectors can be derived directly from the local rules that define the task to be accomplished by the network. The sets contain desired and forbidden input/output combinations, resp., which lead to appropriate inequalities bounding the parameter space. Any point in the solution space, i.e. any parameter set that satisfies all inequalities, guarantees the correct operation of the CNN for the given task.

The exact design method has been demonstrated in detail for the problem of edge extraction. A CNN with a parameter set that satisfies the constructed inequalities extracts the edge of any pattern at any position in a two-dimensional layer of arbitrary size, independently of the initial state.

References

[1] Leon O. Chua und Lin Yang. Cellular neural networks: Theory. *IEEE Transactions on Circuits and Systems*, 35(10):1257–1272, October 1988.

[2] Leon O. Chua und Lin Yang. Cellular neural networks: Applications. *IEEE Transactions on Circuits and Systems*, 35(10):1273–1290, October 1988.

[3] Fan Zou und Josef A. Nossek. Design method for Cellular Neural Network with linear relaxation. In *IEEE International Symposium On Circuits And Systems*, pages 1323–1326, Singapore, June 1991.

[4] Leon O. Chua und Patrick Thiran. An Analytic Method for Designing Simple Cellular Neural Networks. *IEEE Transactions on Circuits and Systems*, 38(11):1332–1341, November 1991.

Cellular Neural Network Design with continuous Signals

Stephan Schwarz and Wolfgang Mathis
University of Wuppertal
Department of Electrical Engineering
Institute for Computer Aided Circuit Design
Fuhlrottstraße 10, D-5600 Wuppertal 1, Germany,
Tel. xx49-202-4393008 and Fax. xx49-202-4393040.

Abstract

In this paper are indroduced basic design methods for the class of cellular neural networks (CNN' s) with continuous input signals. The realistic Modell of CNN' s are proposed by Chua and Yang [2, 3] and combine components of the Hopfield-Net [8], the Cellular Automatas [13] und der Cellular Systems [14]. The CNN design methods integrate special conditions for technical architectures with respect to real-time implementations. Hačijan's polynomial solution method [5] are applied to solve the set of linear inequalities which correspond with the CNN design.

1 Introduction

A CNN architecture for image processing are marked by regular arrangement of identical cells, which are shown as rectangular 5×8 CNN with 40 cells c^k for $1 \leq k \leq 40$ left in figure 1. All cells characterize an identical implementation, that is chosen as a discret-time system [1, 4, 6, 11] and follows as signal flow graph right in figure 1.

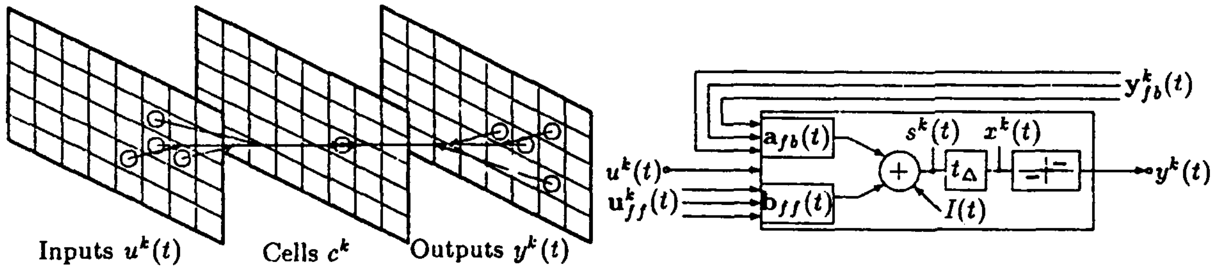


Figure 1: Arrangement of cells and signal flow graph

All cells performe weighted sums of local in- and output signals which delayed by a discrete-time t_Δ and pass the results through nonlinearities into the new output signals. Figure 2 shows all vector elementes of input, output and weight connections, which are identical constructed around all cells c^k.

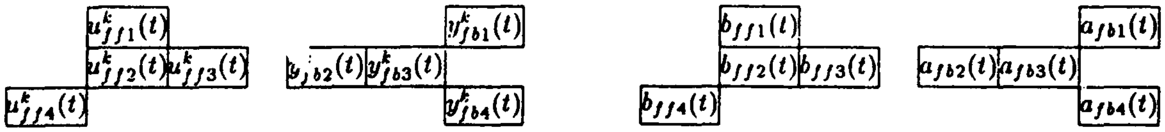


Figure 2: Arrangement of vector elementes

The following table 1 presents the denominations of all quantities.

$IM = \{-1, 0, +1\} := \{\text{white , grey, black}\}$,	$O \in IN$: feedback connections,
$P \in IN$: feedforward connections,	$u^k(t) \in [-1, +1]$: input signal of c^k,
$y^k(t) \in IM$: output signal of c^k,	$\mathbf{y}^k_{fb}(t) \in IM^P$: feedback vector of c^k,
$\mathbf{u}^k_{ff}(t) \in [-1, +1]^O$: feedforward vector of c^k,	$\mathbf{v}^{k\mathsf{T}}(t) := (\mathbf{y}^{k\mathsf{T}}_{fb}(t), \mathbf{u}^{k\mathsf{T}}_{ff}(t), 1) \in IR^{O+P+1}$,
$\mathbf{a}_{fb}(t) \in IR^P$: feedback weights,	$\mathbf{b}_{ff}(t) \in IR^O$: feedforward weights,
$I(t) \in IR$: Source and	$\mathbf{w}^{\mathsf{T}}(t) := (\mathbf{a}^{\mathsf{T}}_{fb}(t), \mathbf{b}^{\mathsf{T}}_{ff}(t), I(t)) \in IR^{O+P+1}$.

Table 1: The denominations of all quantities

The positions of the vector elements for the in-, outputs and the constant 1 show figure 2 and table 1. The same form for the temporal variable weights or "Cloning Template" of the in-, outputs and source follows also in figure 2 and table 1, which is identical for all cells. The equations (1, 2, 3 and 4) represent the mathematical description of the dynamical behaviour of all cells in all discrete-times t with $t \in \{0, 1t_\Delta, 2t_\Delta \ldots\}, t_\Delta \in IR_+$. The sums $s^k(t)$ of the weighted in- and outputs and source

$$s^k(t) = \mathbf{a}^{\mathsf{T}}_{fb}(t)\mathbf{y}^k_{fb}(t) + \mathbf{b}^{\mathsf{T}}_{ff}(t)\mathbf{u}^k_{ff}(t) + I(t) \tag{1}$$

are transformed with the vectors of in- and outputs $\mathbf{v}^k(t)$ and weights $\mathbf{w}(t)$ in the sums

$$s^k(t) = \mathbf{w}^{\mathsf{T}}(t)\mathbf{v}^k(t). \tag{2}$$

The delayed step t_Δ of the sums are noted down with the states $x^k(t)$ in equation (3).

$$x^k(t) = s^k(t - t_\Delta) \tag{3}$$

The nonlinearities perform the states $x^k(t)$ into the outputs and are shown in figure 3.

$$y^k(t) = \begin{cases} -1 & \forall \quad -3 < x^k(t) < -1 \\ 0 & \forall \quad -1 < x^k(t) < +1 \\ 1 & \forall \quad +1 < x^k(t) < +3. \end{cases} \tag{4}$$

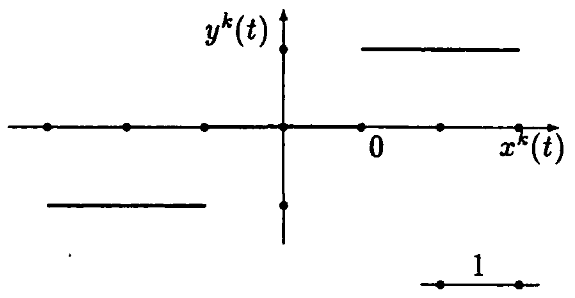


Figure 3: Stepped nonlinearity

The iteration equation (5) arise out of the equations(3 and 4), which calculates the present outputs with the past in- and outputs and cloning templats.

$$y^k(t) = \begin{cases} -1 & \forall \quad -3 < s^k(t - t_\Delta) < -1 \\ 0 & \forall \quad -1 < s^k(t - t_\Delta) < +1 \\ +1 & \forall \quad +1 < s^k(t - t_\Delta) < +3. \end{cases} \tag{5}$$

The bounds of the outputs [9] follow with table 1 und equation (4)

$$\forall \quad \mathbf{v}^k(t - t_\Delta) \in IR^{O+P+1} \quad \Rightarrow \quad y^k(t) \in IM. \tag{6}$$

The CNN' s perform out of all initial values $u^k(0), y^k(0), \mathbf{w}(0)$ with all sequences of operations $\mathbf{w}(t)$ and inputs $u^k(t)$ a sequence of outputs $y^k(t)$.

2 Design

The application of CNN' s for image processing are determined by the CNN design methods for the determination [7, 10, 11, 12, 15] of the temporal sequence of local operations $\mathbf{w}(t)$. All values of weights $w_i(t)$, sums $s^k(t)$ and states $x^k(t)$ are restricted into intervals.

$$|w_i(t)| \leq 2, \quad |s^k(t)| = |x^k(t)| \leq 3. \tag{7}$$

The start points of the design form the explicite iteration equation (5) and equation (2) which characterize the relation between the identical local connected inputs $u^k(t)$, outputs $y^k(t)$ and the cloning templates $\mathbf{w}(t)$.

For a given temporal sequence of in- and outputs inspects the following design method the existence of the sequence of cloning templates which perform the given sequence of outputs. For this purpose are estimated in equation (5) the linear sums $s^k(t-t_\Delta)$, look at the design method (8), between their extreme points with respect to the following output $y^k(t)$ by linear inequalities for all cells.

$$\begin{array}{ll}
\textit{if} \quad y^k(t) = -1 \Rightarrow -3 < s^k(t-t_\Delta) < -1 & \\
\textit{then} & \\
\quad \mathbf{v}^{k\mathsf{T}}(t-t_\Delta)\mathbf{w}(t-t_\Delta) > -3 \quad \mathbf{v}^{k\mathsf{T}}(t-t_\Delta)\mathbf{w}(t-t_\Delta) < -1 & \\
 & \\
\textit{if} \quad y^k(t) = 0 \Rightarrow -1 < s^k(t-t_\Delta) < +1 & \\
\textit{then} & (8) \\
\quad \mathbf{v}^{k\mathsf{T}}(t-t_\Delta)\mathbf{w}(t-t_\Delta) > -1 \quad \mathbf{v}^{k\mathsf{T}}(t-t_\Delta)\mathbf{w}(t-t_\Delta) < +1 & \\
 & \\
\textit{if} \quad y^k(t) = +1 \Rightarrow +1 < s^k(t-t_\Delta) < +3 & \\
\textit{then} & \\
\quad \mathbf{v}^{k\mathsf{T}}(t-t_\Delta)\mathbf{w}(t-t_\Delta) > +1 \quad \mathbf{v}^{k\mathsf{T}}(t-t_\Delta)\mathbf{w}(t-t_\Delta) < +3 &
\end{array}$$

The further technique are introduced by a visual example. The detection of black-white differences in images can be solved by a feedforward and a feedback connection $\mathbf{v}^{k\mathsf{T}}(t) = (u^k(t), y^k(t))$. The two images are assigned to the inputs $u^k(t-t_\Delta)$ and outputs $y^k(t-t_\Delta)$ and the differences are assigned to the next outputs $y^k(t)$. Figure 4 shows all transients and Table 2 shows their values.

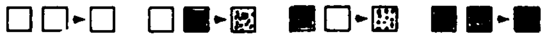


Figure 4: Transients

	$\mathbf{v}^{k\mathsf{T}}(t-1)$	$y^k(t)$		$\mathbf{v}^{k\mathsf{T}}(t-1)$	$y^k(t)$
1	$(-1,-1)$	-1	3	$(+1,-1)$	0
2	$(-1,+1)$	0	4	$(+1,+1)$	$+1$

Table 2: In- und output values of differences

The design methods (8) and restrictions (7) lead to the inequalities (9) with the set of solutions in figure 8.

$$\begin{array}{l}
+w_1 + w_2 > +1 \\
+w_1 - w_2 > -1 \\
\mp w_1 - w_2 > +1
\end{array} \tag{9}$$

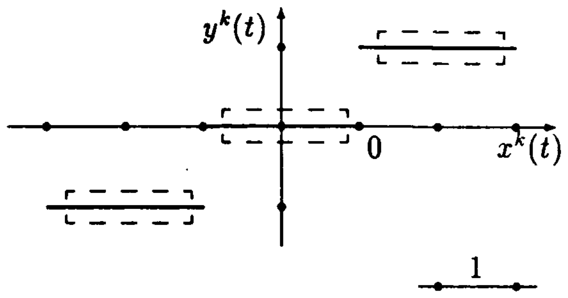


Figure 5

The intervals of input, output and sum values $\mathbf{v}_\Delta$, y_Δ and s_Δ, see figure 5 , are shown in equation (10) for $y^k(t) = 0$.

$$\begin{array}{l} if \quad y^k(t) = 0 \pm y_\Delta \quad \Rightarrow \quad -1 + s_\Delta < s^k(t - t_\Delta) < +1 - s_\Delta \\ then \\ (\mathbf{v}^{k\mathsf{T}}(t - t_\Delta) \pm \mathbf{v}_\Delta)\mathbf{w}(t - t_\Delta) > -1 + s_\Delta, \quad (\mathbf{v}^{k\mathsf{T}}(t - t_\Delta) \pm \mathbf{v}_\Delta)\mathbf{w}(t - t_\Delta) < +1 - s_\Delta \end{array} \tag{10}$$

All combinations of values, see figure 6, lead to a continuous set of linear inequalities in figure 7. This set of solutions can be estimated by the finit number of extreme inequalities. The number is 2^{O+P+1} and the combinations follows with a binary system from 0 to 2^{O+P}.

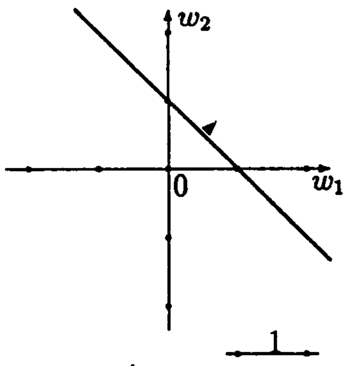


Figure 6

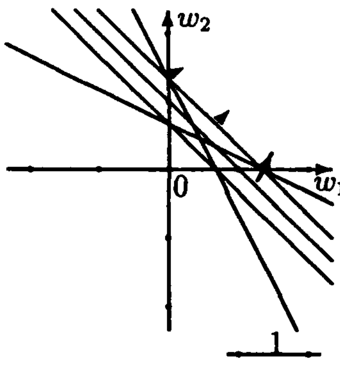


Figure 7

Figure 9 shows the solutions of (9) which are estimated by the inequalities of figure 7.

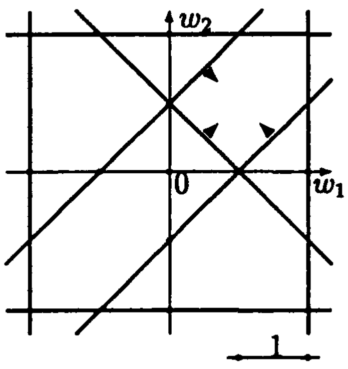


Figure 8

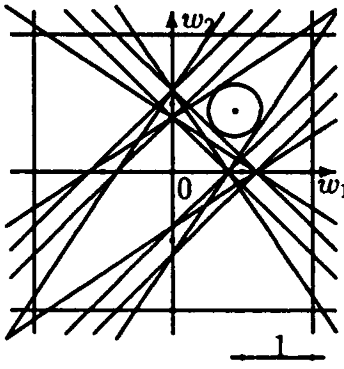


Figure 9

A stabil technical implementation [10] of realistic cloning template leads to a central point of the maximal ball with radius r within the set of solutions in figure 9. This cloning template is stabil for all defects of the cloning template which are lower than the maximal radius.

$$\mathbf{v}^{\mathsf{T}}\mathbf{w} + \zeta + \mid \mathbf{v} \mid r \geq 0, \quad r \in IR \tag{11}$$

For the determination of the maximal ball and radius are introduced parameter [10 and 11] in all inequatities which are present in equation (11, 12).

$$\begin{gathered} \mathbf{v}^{k\mathsf{T}}(t - t_{\Delta})\mathbf{w}(t - t_{\Delta}) + \zeta^k_{>} + \mid \mathbf{v}^k \mid r \geq 0 \\ \zeta^k_{>}, \zeta^k_{<} \in IR \end{gathered} \tag{12}$$

All inequalities of discrete step for a parameter build a set of inequalities (13), which can be solved by Hačijan's polynomial solution method [5].

$$\alpha_\pi^{\mathsf{T}} \chi^\sigma < \beta_\pi \tag{13}$$

The polynomial solution method constructs a sequence σ of ellipses which approximate the set of solutions and present a solution by the characteristic central point χ^σ and tranformation matrix Θ^σ. The method are denoted in the following

```
step 0: $\sigma = 0, \chi^\sigma = \mathbf{0}, \Theta^\sigma = (O + P + 1)2^2\mathbf{E}$
step 1: if $\forall \pi \in \{1, \ldots, \Pi\} \alpha_\pi^{\mathsf{T}} \chi^\sigma < \beta_\pi$
        then stop with solution $\chi^\sigma$
        else $\alpha_\pi^{\mathsf{T}} \chi^\sigma > \beta_\pi$
        $\omega_\pi = \Theta^\sigma \alpha_\pi, \delta_\pi = \sqrt{\alpha_\pi^{\mathsf{T}} \Theta^\sigma \alpha_\pi}, \tilde{\beta}_\sigma = -\frac{\alpha_\pi^{\mathsf{T}} \chi^\sigma - \beta_\pi}{\delta_\pi}$
step 2: $\chi^{\sigma+1} = \chi^\sigma - \frac{(1-\lambda\tilde{\beta}_\sigma)\omega_\pi}{(\lambda+1)\delta_\pi}, \Theta^{\sigma+1} = \frac{\lambda^2(1-\tilde{\beta}_\sigma)}{\lambda^2-1}[\Theta^\sigma - \frac{2(1-\lambda\tilde{\beta}_\sigma)\omega_\pi\omega_\pi^{\mathsf{T}}}{(\lambda+1)(1-\tilde{\beta}_\sigma)\delta_\pi^2}]$
        $\sigma = \sigma + 1$ goto step 1
$\pi, \Pi, \sigma, \lambda \in IN; \beta_\pi, \delta_\pi, \tilde{\beta}_\sigma \in IR; \alpha_\pi, \chi^\sigma, \omega_\pi \in IR^\lambda; \Theta^\sigma \in IR^{\lambda \times \lambda}$.
```

If a solution exist for every discrete step t in all sets of inequalities (12 and 13) then all solutions build the sequence of cloning templats $\mathbf{w}(t)$.

References

[1] Baldi P. : *Neural Networks, Acyclic Orientations of the Hypercube, and Sets of Orthogonal Vectors, SIAM J. DISC. MATH. Vol. 1, No. 1, February 1988.*

[2] Chua L. O. and Yang L. : *Cellular Neural Networks: Theory, IEEE Transactions on Circuits and Systems, Vol. 35, No. 10, October 1988.*

[3] Chua L. O. and Yang L. : *Cellular Neural Networks: Applications, IEEE Transactions on Circuits and Systems, Vol. 35, No. 10, October 1988.*

[4] Frühauf N. and Lüder E. : *Realization of CNN's by Optical Parallel Processing with Spatial Light Valves, IEEE 1. International Workshop on Cellular Neural Networks and their Applications, Budapest (Hungary), December 16-19th, 1990, ISBN 951-721-239-9.*

[5] Hačijan L. G. : *A POLYNOMIAL ALGORITHM IN LINEAR PROGRAMMING, Soviet Mathematics Doklady, Vol. 20, No. 5, October 1979.*

[6] Harrer H. and Nossek J. A. : *Discrete-Time Cellular Neural Networks, Technical University Munich (Germany), Institute for Network Theory and Circuit Design, Report No: TUM-LNS-TR-91-7, March 1991.*

[7] Jiang X. C. , Hegde M. and Naraghi-Pour M. : *Neural Network Design Using Linear Programming and Relaxation, Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA 70803, LEQSF-RD-A-5.*

[8] Lippmann R. P. : *An Introduction to Computing with Neural Nets, IEEE ASSP Magazine April 1987.*

[9] Oppenheim A. V. and Schafer R. W. : *DISCRETE-TIME SIGNAL PROCESSING, PRENTICE HALL 1989, ISBN 0-13-216771-9.*

[10] Schwarz S. and Mathis W. : *A Design Algorithm for Cellular Neural Networks, IEEE 2. International Conference on Microelectronics for Neural Networks, Munich (Germany), October 16-18th, 1991.*

[11] Schwarz S. and Mathis W. : *Theory of Cellular Neural Network Design, ISSSE International Symposium on Signals, Systems and Electronics, Paris, September 1-4th, 1992.*

[12] Vandenberghe L., Tan S. and Vandewalle J. : *Cellular Neural Networks: Dynamic Properties and Adaptive Learning Algorithm, Neural Networks, EURASIP Workshop 1990, Sesimbra, Portugal, February 1990, Springer-Verlag, ISBN 3-540-52255-7.*

[13] Wolfram S. (Eds.): *Theory and Applications of Cellular Automata, New York: World Scientific 1986.*

[14] Wunsch G. and Mathis W. : *Tcward a Theory of Cellular Systems, IEEE 1. International Workshop on Cellular Neural Networks and their Applications, Budapest (Germany), December 16-19th, 1990, ISBN 951-721-239-9.*

[15] Zou F. , Schwarz S. and Nossek J. A. : *Cellular Neural Networks Design Using a Learning Algorithm, IEEE 1. International Workshop on Cellular Neural Networks and their Applications, Budapest (Hungary), December 16-19th, 1990, ISBN 951-721-239-9.*

DESIGNING DISCRETE-TIME CELLULAR NEURAL NETWORKS FOR THE EVALUATION OF LOCAL BOOLEAN FUNCTIONS

ZBIGNIEW GALIAS
University of Mining and Metallurgy
IMISUE, Department of Electrical Engineering,
al.Mickiewicza 30, 30-059 Cracow, Poland
Tel. (+48 12) 33 91 00 ext. 3613, Fax (+48 12) 33 10 14

Abstract

This paper describes general methods of designing a discrete-time cellular neural network implementing an arbitrary Boolean function defined on the r-neighbourhood. This is achieved by operating the network with time-variant templates as a cellular automaton that processes only binary inputs. These methods are suital le for solving local tasks. As an example, testing minimal distances is discussed.

1. INTRODUCTION

Discrete-time cellular neural network as introduced in [3] is a large scale nonlinear circuit, composed of locally connected cells. It has a simple nonlinear circuit at each node and is very well suited for VLSI implementations.

In this paper we restrict ourselves to binary valued operations and study how to implement local Boolean functions using DTCNNs. We prove that every Boolean function defined on the r-neighbourhood can be implemented in a single layer DTCNN with time-variant templates. This is achieved by operating the network as a cellular automaton that processes only binary inputs. General methods for designing DTCNNs implementing Boolean functions are presented. They are based on a digital convolution. In every iteration step the output state of any cell can be changed only if the input values within the r-neighbourhood match exactly the given pattern, stored in the control operator. The coefficients of control operator are restricted to the set {-1,0,1}, while the feedback operator consists of self-feedback only.

Presented methods are suitable ior solving local tasks. In the second part of the paper a new application of DTCNN is described. Testing distances between objects for an arbitrary value of the minimal distance is performed by a two-layer network with time-variant templates.

2. NOTATIONS

Definition 1: *Discrete-time cellular neural network* (DTCNN) using time-variant templates is defined by the following recursive algorithm:

for k>0

$$x^c(k) = a_d^c(k)\cdot y^d(k-1) + b_d^c(k)\cdot u^d + i^c(k) \tag{1}$$

$$y^c(k) = f(x^c(k)) = \begin{cases} 1 \text{ for } x^c(k)\geq 0 \\ -1 \text{ for } x^c(k)<0 \end{cases} \tag{2}$$

In (1) the Einstein summation convention [7] is used. For its convenience it will be used throughout the paper.

The feedback coefficients $a^c_d(k)$, the control coefficients $b^c_d(k)$ and the thresholds $i^c(k)$ are real parameters. The initial state $y^c(0)$, the output state after the kth iteration $y^c(k)$ and the input u^c of a cell c are binary valued.

The cells of the network form a regular grid. The cells are only locally connected. By the definition the template coefficients are translationally invariant. A single template for given k is called *a snapshot template.*

This definition differs from the one given in [2] in that the input values u^c are assumed to be constant, while originally they have been time-variant.

Let r be a constant integer value. Let us consider a cell c. *The r-neighbourhood* of the cell c is defined as the set of all cells within the distance r, with respect to the 'maximum' metric. It is denoted by $N_r(c)$.

Let M=M(r) be the cardinality of $N_r(c)$. For each cell its r-neighbourhood contains $M(r)=(2r+1)^2$ cells, when restricting to square grids.

Let $d_1, d_2, \dots, d_M$ denote the cells belonging to $N_r(c)$. $U_c := (u^1, \dots, u^M)$ is the vector of the inputs of all cells within the r-neighbourhood of the cell c.

Let S be the set of all possible binary input patterns within $N_r(c)$:

$$S = \{u=(u^1, \dots u^M) \in \{-1,+1\}^M\} \tag{3}$$

Definition 2: Let g be an arbitrary Boolean function defined on S:

$$g: S \longrightarrow \{-1,+1\}$$

where +1 is interpreted as *true* and -1 as *false*.
We say that a DTCNN *implements* function g if there exists $k_0 \geq 0$ such that

$$\forall c \ \forall U_c \in S \ \forall k \geq k_0 \quad y^c(k) = g(U_c)$$

3. DESIGNING OF DTCNN PERFORMING BOOLEAN FUNCTIONS

Two methods of designing DTCNNs implementing an arbitrary Boolean function can be proposed.

Method 1: In order to implement an arbitrary Boolean function perform the following two steps:

Step 1: Let us write the Boolean function g in a sum-of-products form:

$$g(u^1, \dots, u^M) = (u^{1,1} \wedge \dots \wedge u^{1,w(1)}) \vee \dots \vee (u^{n,1} \wedge \dots \wedge u^{n,w(n)}) \tag{4}$$

We have n products. The kth product has w(k) factors.
$u^{k,j}$ denotes one of the variables $u^1, \dots u^M$, which may be negated.

Step 2: Let n be the number of snapshot templates.

$\forall c \ \forall k \in \{1, \dots, n\}$

$y^c(0)=-1$, $i^c(k)=+1$,

$a^c_c(k)=w(k)$, $a^c_d(k)=0$ for $c\neq d$,

for $d=d_s\in N_r(c)=\{d_1,d_2,\dots,d_M\}$:

$b^c_d(k)=+1$ if u^s is present in the kth product of (4)

$b^c_d(k)=-1$ if $\overline{u^s}$ is present in the kth product

$b^c_d(k)=0$ if u^s and $\overline{u^s}$ are not present in the kth product

for $d\notin N_r(c)$: $b^c_d(k)=0$.

The binary valued input pattern is the information to be processed. n iterations with time-variant templates should be performed. Then we can either stop processing or we can continue with the last snapshot template.

Proof of the correctness of method 1:
First let us consider an arbitrary cell c, any integer $k\in\{1,\dots,n\}$ and several cases depending on the value of $y^c(k-1)$ and the input vector **u**.

Case 1: $y^c(k-1)=+1$

$$a^c_d(k)\cdot y^d(k-1) + i^c(k) = w_k + 1$$

Because exactly w_k elements of $b^c_d(k)$ (for given c) are equal to +1 or -1 and other elements are 0 then:

$$|b^c_d(k)\cdot u^d| \leq |b^c_d(k)|\cdot 1^d = w_k$$

Hence $x^c(k)>0$ and $y^c(k)=+1$.

Case 2: $y^c(k-1)=-1$ and $\forall d$: $b^c_d(k)=u^d$ or $b^c_d(k)=0$

$$a^c_d(k)\cdot y^d(k-1) + i^c(k) = -w_k + 1$$

$$b^c_d(k)\cdot u^d = |b^c_d(k)|\cdot 1^d = w_k$$

Hence $y^c(k)=f(-w_k+1+w_k) = +1$.

Case 3: $y^c(k-1)=-1$ and $\exists d$: $u^d\neq b^c_d(k)\neq 0$

$$b^c_d(k)\cdot u^d \leq |b^c_d(k)|\cdot 1^d - 2 = w_k - 2$$

$$x^c(k) = a^c_d(k)\cdot y^d(k-1) + i^c(k) + b^c_d(k)\cdot u^d \leq -w_k + 1 + w_k - 2 = -1$$

In this case $y^c(k)=-1$.

It effects from cases 1-3 that a transition from +1 to -1 is not allowed and a transition from -1 to +1 occurs in step k only if the input pattern in the r-neighbourhood exactly matches the non-zero values of the matrix b(k). The function g is written in the sum-of-products form and every iteration step computes one product of this sum.

As $y^c(0)=-1$ then $y^c(n)=+1$ if and only if $\exists k\in\{1,\dots,n\}$ such that

$$\forall d : b^c_d(k)=u^d \text{ or } b^c_d(k)=0.$$

Because for every $k\geq n$ $y^c(k)=y^c(n)$, the function g is implemented properly.

Method 2: Let us write function g in a product-of-sums form:

$$g(u^1,\dots,u^M) = (u^{1,1}\vee\dots\vee u^{1,w(1)})\wedge\dots\wedge(u^{n,1}\vee\dots\vee u^{n,w(n)}) \tag{5}$$

All parameters of the network should be the same as in method 1 with the following differences: $i^c(k)=-1$ (instead of +1), $y^c(0)=+1$ (instead of -1).

Proof for method 2 is dual to that of method 1.

Remark: Both methods can be used for the implementation of an arbitrary Boolean function. This can be seen by noting that any Boolean function can be written in both the sum-of-products and the product-of-sums forms [4]. The methods 1 and 2 usually solve the same problem with the different complexity.

Example: We want to detect horizontal and vertical black lines of length greater than two pixels. This is equivalent to detecting all black pixels which have their horizontally adjacent pixels black or which have their vertically adjacent pixels black.

This function can be presented in the sum-of-products form as:

$$g(u^1,\dots,u^9) = (u^2 \wedge u^5 \wedge u^8) \vee (u^4 \wedge u^5 \wedge u^6)$$

Method 1 leeds to the following solution: n=2, $\forall c \;\; y^c(0)=-1$, u=image,

$$a(1)=a(2)=\boxed{4}, \; b(1)=\begin{array}{|ccc|}\hline 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ \hline\end{array}, \; b(2)=\begin{array}{|ccc|}\hline 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ \hline\end{array}, \; i(1)=i(2)=+1$$

4. TESTING MINIMAL DISTANCES

The task is to detect black objects, which are closer then a distance of k pixels, where k is a given constant integer value. The solution of this problem can be used in layout design for checking if any two objects are too close to each other.

The solution for k=2 described in [5] is based on extracting black pixels having a white neighbour in a given direction and a black second neighbour in the same direction. It can be done with 2-neighbourhood templates. This solution allows to extract pixels in one direction. To be able to solve this problem for four directions it was proposed to process simultaneously but separately in four direction and then perform the OR function on the outputs.

In this paper we use DTCNN architecture with time-variant templates for solving the problem for an arbitrary value of minimal distance. Each snapshot template is equivalent to one layer of hardware [5] while the OR function is performed at the same time by choosing templates properly.

Before the solution of that problem is given some definitions have to be introduced.

Let the distance between two pixels w, z be defined by the 'maximum' metric:

$$d(w,z):=\max\{|w_1-z_1|, |w_2-z_2|\}$$

where w_i, z_i are the integer coordinates of the pixels.

Definition 3: Two black pixels w and z are called *k-separated*, if their distance is k and they cannot be connected by a black four-connected path completely enclosed in the rectangle determined by w and z, where:

- the *four-connected path* is a sequence of orthogonally adjacent pixels (such pixels are also called *four-connected pixels* [2]),

∘ the *rectangle determined by* w *and* z is the minimal rectangle containing w and z with horizontal and vertical sides.

Detection of 1-separated pixels

The task is to find all 1-separated pixels and mark their positions on the grid. The solution is based on searching for white pixels which have 1-separated black neighbours. This problem can be represented in the form of a Boolean function as:

$$f(u^1,\dots u^9)=(\overline{u^5}\wedge u^6\wedge u^8\wedge \overline{u^9})\vee(\overline{u^5}\wedge u^2\wedge u^6\wedge \overline{u^3})\vee(\overline{u^5}\wedge u^2\wedge u^4\wedge \overline{u^1})\vee(\overline{u^5}\wedge u^4\wedge u^8\wedge \overline{u^7})$$

With method 1, we can construct the following DTCNN:

n=4, y(0)=-1, u=image, **a**(k)=4, **i**(k)=1 for every k∈{1..4}

$$\mathbf{b}(1)=\boxed{\begin{matrix}0&0&0\\0&-1&1\\0&1&-1\end{matrix}},\ \mathbf{b}(2)=\boxed{\begin{matrix}0&1&-1\\0&-1&1\\0&0&0\end{matrix}},\ \mathbf{b}(3)=\boxed{\begin{matrix}-1&1&0\\1&-1&0\\0&0&0\end{matrix}},\ \mathbf{b}(4)=\boxed{\begin{matrix}0&0&0\\1&-1&0\\-1&1&0\end{matrix}}$$

In this solution each detected pair of 1-separated pixels is marked by two black pixels in the output. We can modify the network in the following way: n=2, the templates for k=1,2 are the same and still all 1-separated pixels are detected.

Similar networks can be used for detection of 2 and 3-separated pixels. When searching for 2-separated pixels we need 8 snapshot templates and 1-neighbourhood. For detection of 3-separated pixels we need 12 snapshot templates and 2-neighbourhood of matrix **b**.

Detection of k-separated pixels

We show how to use the solution for k=2,3 to build a 2-layer DTCNN which detects k-separated pixels for every k. The idea is as follows:

```
j:=2;
repeat
  detect 2-separated pixels;{equivalent to detection of j-separated pixels}
  detect 3-separated pixels;{equiv. to detection of (j+1)-separated pixels}
  increase objects by one pixel in every direction;
until (j>k);
```

Increasing objects rely on converting white pixels to black if any of their eight-connected neighbours is black. The DTCNN performing this task is described in [1]. Increasing objects can reduce the distance between objects by two, then we are sure that no too distant objects are joined and we do not lose any k-separated pixels, because earlier we have checked the problem for k=2,3.

The described algorithm can be realized in a 2-layer DTCNN. The first layer has a constant template for increasing objects. The second one is a layer with cyclic templates (the cycle length being 20). The first 8 snapshot templates solve the problem for k=2, the next 12 for k=3. The first layer should be clocked 20 times slower then the second one and the clocks must have different phases. Computation of the result for the first layer must be completed before computation in the second layer starts. To avoid this and make the timing requirements not so strong we can add one snapshot template to the second layer which does not change the output states of the cells (**a**=1, **b**=0, **i**=0) and during this 21st iteration perform one iteration in the first layer.

5. CONCLUSIONS

Using presented methods we are capable of implementing any local Boolean function defined on the r-neighbourhood. For that we need a single-layer DTCNN with time-variant templates, where control operator is defined on the r-neighbourhood and the feedback operator consists of self-feedback only. We are able to realize both the sum-of-products (method 1) and the product-of-sums (method 2) form of any Boolean function, and we choose the shorter one. These methods can be used for hexagonal grids without any modifications.
Several examples are demonstrated and discussed. By means of method 1 the problem of testing minimal distances is solved. The solutions for the case of the 'maximum' metric and k=1,2,3 are described. For that the single layer CNN with time-variant templates is required. A two-layer structure, which solves that problem for arbitrary value of k, is also described. For that application the 2-neighbourhood of the control operator and the 0-neighbourhood of the feedback operator are sufficient. Similar networks can be used, when searching for white pixels k-separated by black ones. The solution for that problem can be also utilized in layout design for checking is there are any too thin black objects on the grid. The solution for the Manhattan metric has also been found.

ACKNOWLEDGMENTS

This work was supported in part by the Polish Committee of Scientific Research, grant no.: 30678 91 01.

The author would like to thank professor Josef A.Nossek for organizing the research stay at the Technical University in Munich and for useful discussions. Special credits deserve Hubert Harrer for active help during preparation of this paper.

REFERENCES

[1] Hubert Harrer, J.A.Nossek, "Time-Discrete Cellular Neural Networks", Institute for Network Theory, TU Munich, Rep. No.: TUM-LNS-TR-91-7.
[2] Hubert Harrer, J.A.Nossek, "Multilayer Discrete-Time Cellular Neural Networks Using Time-Variant Templates", Institute for Network Theory, TU Munich, Rep. No.: TUM-LNS-TR-91-7.
[3] L.O.Chua, B.E.Shi, "Exploiting Cellular Automata in the Design of Cellular Neural Networks for Binary Image Processing", Electronics Research Laboratory, University of California, Berkeley, Memorandum No. UCB/ERL M89/130, 15 November 1989.
[4] J.D.Greenfield, "Practical Digital Design Using IC's", 2nd ed., New York: Wiley, 1983.
[5] T.Boroz, K.Lotz, A.Radvanyi, T.Roska, "Some Useful New, Nonlinear and Delay-Type Templates", Computer and Automation Inst., Hungarian Academy of Sciences, Rep.No.: DNS-1-1991.
[6] K.Preston, Jr. and M.J.B.Duff, "Modern Cellular Automata: Theory and Applications", Plenum Press, New York, 1984.
[7] J.A.Nossek, G.Seiler, T.Roska, L.O.Chua, "Cellular Neural Networks: Theory and Circuit Design", Institute for Network Theory, TU Munich, Rep.No.: TUM-LNS-TR-90-7.

Template Synthesis of Cellular Neural Networks for Information Coding and Decoding

Mamoru Tanaka[*] K. R. Crounse[†] Tamas Roska [‡]

Abstract

This paper describes the use of analog Cellular Neural Networks (CNNs) for information coding and decoding – especially for the case of moving images. The dynamics of the coding (C-) and decoding (D-) CNNs are described by generalized CNN state equations. The C-CNN encodes the image by *structural compression* and *halftoning*. The D-CNN decodes the received data through a *reconstruction* process so as to almost recognize the original input to the C-CNN. The importance of our compression and quantization technique lies in the ability to make the computations with only local connections.

A *dynamic quantization* is performed in the C-CNN to decide the binary value of each pixel from the neighboring values. In order to reduce the error between the original gray image and reconstructed halftone image, the template synthesis problem is addressed from the viewpoint of energy minimization.

The *structural compression* template synthesis problem is discussed from the viewpoints of topological and regularization theories. The structurally compressed image is regenerated in the D-CNN by a *dynamic current distribution*.

The communication system in which the C- and D-CNNs are embedded consists of a differential transmitter with an internal receiver model in the feedback loop. Examples of the performance of the complete system are given.

1 Introduction

The *Cellular Neural Network* (CNN) first proposed by L. O. Chua and L. Yang [1] [2] is a locally interconnected, large-scale nonlinear analog neural network which has applications to many practical problems. A generalization of the CNN paradigm was made by T. Roska and L. O. Chua which covers a broad class of programmable multidimensional cellular analog processing arrays [3] [4]. Of special interest here, it has been shown that the nonlinear dynamics of the analog CNN can perform *image halftoning* [5].

1.1 System Summary

This paper describes the use of the CNN for data coding and decoding for the purposes of lossy fixed-rate compression and quantization. In a communication system, the CNN would be used in both the transmitter and receiver. It will be shown how the technique can be applied to the transmission of both still and moving images.

Two distinct compression techniques were developed which only require local connections for computation. As shown in the Figure 1, the coding (C-) and decoding (D-) CNNs implement these encoding-decoding pairs.

Two types of compression are performed in the C-CNN: *structural compression* (S) and *dynamic quantization* (H). The C-CNN first performs the structural compression which locally maps the input gray image to a smaller gray image. The main work of the C-CNN is a *dynamic quantization* by which the gray image is mapped to a binary image of the same size for digital transmission. The dynamic quantization is done by halftoning.

At the receiving end, the D-CNN attempts to regenerate the original image as closely as possible. The inverse halftoning operation ($\tilde{\mathbf{H}}^{-1}$) converts the received binary image back to gray scale. The decoding

[*] Dept. of Electronics and Electrical Engr., Sophia Univ., Tokyo
[†] Dept. of Electrical Engineering and Computer Science, University of California at Berkeley
[‡] Computer and Automation Institute, Hungarian Academy of Sciences, Uri-u. 49, Budapest, H-1014, Hungary

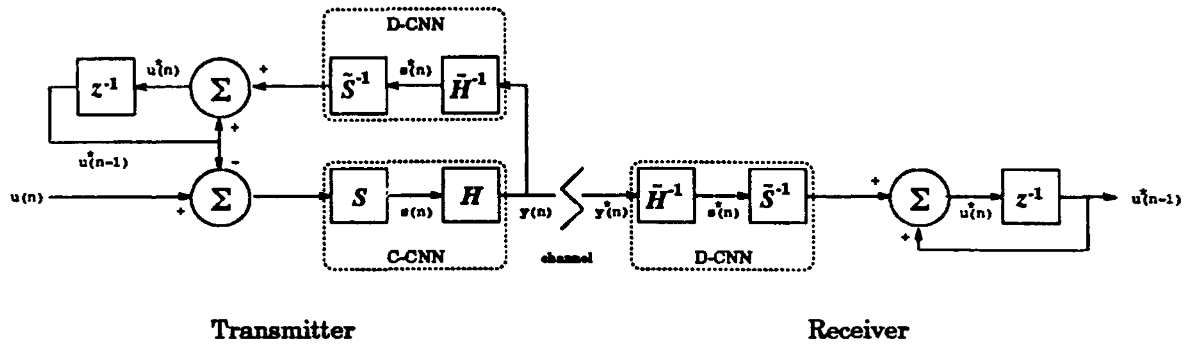


Figure 1: A block diagram of the complete communication system is shown with the functions performed by the C- and D-CNNs. The difference between the current acquired image u(n) and the image on display at the receiver u*($n-1$) is sent to the C-CNN. The C-CNN consists of two functional blocks, the structural compression **S** and the halftoning **H**. Halftoning plays a dual role in compression and quantization of the data for binary transmission. The D-CNN attempts to invert the halftone $\tilde{\mathbf{H}}^{-1}$ and structural compression $\tilde{\mathbf{S}}^{-1}$. Finally, the difference image is added to the one currently on display.

of structural compression ($\tilde{\mathbf{S}}^{-1}$) is done by a *dynamic current distribution* which maps the image back to the original dimensions.

The first compression coding-decoding pair is *structural compression* and the corresponding decoding done by *current regularization*. The structural compression is done in a manner similar to syndrome error coding [6]. The novel current-type CNN is used to perform this compression. The current-type CNN can be written as a special version of the CNN equations.

The second type of compression is *dynamic quantization* which is performed by halftoning the image. Halftoning quantizes each pixel of the image to a binary value. Therefore, halftoning can simultaneously be considered both a form of quantization and fixed-rate image compression. In the usual situation, the 1 bit/pixel will be allocated in a manner that will be a trade-off between gray level and edge preservation. In regions of the image with slowly changing gray levels the algorithm will try to match the spatial density of bits to that of the local gray level. The larger the constant region the more accurately the density can be matched. However, when a small object or fine structure occurs in a region, the bits in that area should be used to preserve its presence. Doing so will be at the expense of accurate preservation of the gray level of the object. Halftoning is therefore a very natural type of compression where the spatial information density is assumed to be constant.

The system in which the C- and D-CNNs are embedded consists of a differential transmitter with a local receiver model in the feedback loop. When transmitting moving images, features in motion are quickly rendered while detail accumulates in stationary areas. When transmitting a still image, the first frame will contain a recognizable image quickly with detail and quantization noise reduction supplied during subsequent transmissions.

In such an arrangement, it can be shown that the error of successive transmissions does not accumulate, in the following sense:

Let $\mathcal{CD}_n = (\tilde{\mathbf{S}}^{-1}\tilde{\mathbf{H}}^{-1})(\mathbf{HS})_n$ be the coding and decoding done on image n. By expanding one level of recursion in the block diagram, we can write

$$\mathbf{u}^*(n+1) = \mathcal{CD}_{n+1}(\mathbf{u}(n+1) - \mathbf{u}^*(n-1) - \mathcal{CD}_n(\mathbf{u}(n) - \mathbf{u}^*(n-1))) + \mathbf{u}^*(n-1) + \mathcal{CD}_n(\mathbf{u}(n) - \mathbf{u}^*(n-1))$$

Now, if the last transmission was perfect, i.e. $\mathcal{CD}_{n+1}$ = Identity, then $\mathbf{u}^*(n+1) = \mathbf{u}(n+1)$ with no accumulated error.

1.2 The CNN

A Cellular Neural Network (CNN), as first proposed in [1], is a continuous time neural network with diameter-limited local interconnections and a unity-gain piecewise linear approximation to the standard sigmoidal output function. The 'neurons' are placed in a regular array and 'synaptic' connections are allowed only locally. The local nature of the interconnections are critical when considering VLSI implementation.

It is convenient for us to write the state equations in convolution form [1]: Let $\mathbf{u}, \mathbf{x}, \mathbf{y}$ be the $M \times N$ *input*, *state*, and *output* gray scale images respectively, which are assumed to be zero outside of their support. Let $\mathbf{T}_A$ and $\mathbf{T}_B$ be functions such that $\mathbf{T}_A, \mathbf{T}_B : \mathbb{Z}^2 \to \mathbb{R}$ with $(T_A)_{i,j}, (T_B)_{i,j} = (T_A)_{-i,-j}, (T_B)_{-i,-j}$ and $(T_A)_{i,j}, (T_B)_{i,j} = 0$ for $\|(i,j)\|_\infty \leq r$, some small neighborhood radius. Let I be a real constant. The ***symmetric space-invariant Cellular Neural Network*** can be described by the following state and output equations:

$$\frac{d}{dt}x_{i,j}(t) = -x_{i,j}(t) + (\mathbf{T}_A * \mathbf{y}(t))_{i,j} + (\mathbf{T}_B * \mathbf{u})_{i,j} + I \tag{1}$$

$$y_{i,j}(t) = f(x_{i,j}(t)) \tag{2}$$

where $f : x \mapsto \frac{1}{2}(|x+1| - |x-1|)$, $x \in \mathbb{R}$.

The functions $\mathbf{T}_A$ and $\mathbf{T}_B$ are called ***cloning templates*** and can be considered $2r+1 \times 2r+1$ matrices. The condition $(T_A)_{i,j} = (T_A)_{-i,-j}$ is called template ***symmetry***. An even stronger condition on the templates is ***isotropy***. For isotropy, the template must satisfy $(T_A)_{i,j} = (T_A)_{k,l}$ for (i,j) and (k,l) the same L_2 distance from the origin. This condition implies template symmetry and is useful in halftoning applications.

We will often discuss the CNN first in matrix-vector form [3]. Let $\mathbf{A}$ and $\mathbf{B}$ be $MN \times MN$ matrices representing the convolutions. They will be of the form block Toeplitz with Toeplitz blocks. Let I be a real constant. The Cellular Neural Network can be described by the following state and output equations:

$$\frac{d}{dt}\mathbf{x}(t) = -\mathbf{x}(t) + \mathbf{A}\mathbf{y}(t) + \mathbf{B}\mathbf{u} + I \tag{3}$$

$$\mathbf{y}(t) = f(\mathbf{x}(t)) \tag{4}$$

1.3 Outline of Paper

In Section 2 the theory behind structural compression is explained. In Section 3 the specifics of halftoning and inverse halftoning are explained from the viewpoint of energy minimization. Section 4 puts the two types of compression together into CNN equations. Finally, in Section 5 we present the techniques and results of simulation to demonstrate the performance of the end-to-end system when using moving images.

2 Structural Compression

The first type of local compression we will consider is ***structural compression***. We will attempt to take advantage of spatial redundancy in an image by collapsing the image support onto a smaller grid.

2.1 Compression

To simplify the discussion, consider the transmission of a fixed frame of a still image, that is, $\mathbf{u}^*(n-1) = \mathbf{0}$. To perform structural compression, the input $\mathbf{u}(n)$ is applied to the links of a graph. At each node, the values on the links connected to that node are summed to produce the output $\mathbf{s}(n)$. If l is the number of links and r is the number of nodes in the graph, the compression ratio will be $\frac{r}{l}$.

The degree of compression obtained is determined by the topology of the graph used. Graphs which tile the plane in a regular way are the most convenient. One possibility is the regular square grid, as shown in Figure 2 which has has two links for every node. Another is the hexagonal grid which has three links for every two nodes.

It is straightforward to implement this method with a circuit by only using local connections, as shown in Figure 2. On each link of the graph, a current source is placed representing the input at that location. In practice this may be the current generated by a photosensor. Then, by KCL, the current leaving the node on the output line is the sum of input currents.

Let $\mathbf{S}$ be the $r \times l$ incidence matrix which represents the compression network. Then, the compression can be represented by the linear transformation $\mathbf{s}(n) = \mathbf{S}\mathbf{u}(n)$ where $\mathbf{s}(n)$ is the syndrome that will be sent to the halftoning network.

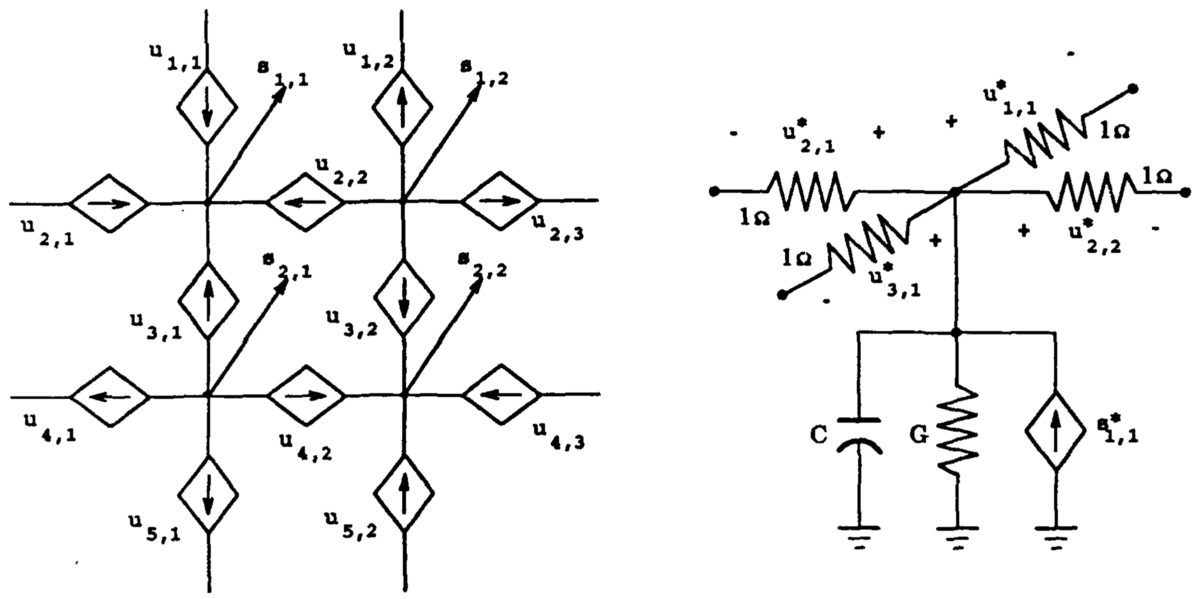


Figure 2: A small section of the encoding (left) and decoding (right) circuits for structural compression. In the encoding network the input is mapped onto the links as current sources. The output current at each node is sent to the halftoning network. One node of the decoding network for dynamic current distribution is shown. Intuitively, the network is reversing the process of encoding by injecting the received currents into the resistive grid. The resistors ensure that the least norm solution is obtained. It is clear that the self conductance G is required since the sum of the received syndrome values will not in general be zero and allows for some current to leak out. The output is the current through the resistors.

2.2 Reconstruction

The problem at the receiving end is to find an appropriate inversion. For this discussion, we assume that the halftoning and transmission are lossless so that $s^*(n) = s(n)$

The reconstruction we will pursue here is the standard pseudo-inverse given by $\mathbf{S}^\mathsf{T}(\mathbf{S}\mathbf{S}^\mathsf{T})^{-1}$. This would give the $\mathbf{u}^*(n)$ with least norm such that $\mathbf{s}^*(n) = \mathbf{S}\mathbf{u}^*(n)$. Unfortunatly, since $\mathbf{S}$ is an incidence matrix, it is singular, having a single zero eigenvalue corresponding to the constant eigenvector.

To correct for the singularity, we define $\tilde{\mathbf{S}}^{-1} = \mathbf{S}^\mathsf{T}(\mathbf{G}+\mathbf{S}\mathbf{S}^\mathsf{T})^{-1}$ where $\mathbf{G}$ is a small multiple of identity. Since $\mathbf{S}\mathbf{S}^\mathsf{T}$ is positive *semi*-definite, $\mathbf{G}+\mathbf{S}\mathbf{S}^\mathsf{T}$ will be positive definite and therefore invertible.

In general, the pseudo-inverse just defined is not represented by a locally connected network. However, consider the feedback network defined by

$$\frac{d}{dt}\mathbf{x}^*(t) = -(\mathbf{G}+\mathbf{S}\mathbf{S}^\mathsf{T})\mathbf{x}^*(t) + \mathbf{s}^*(n) \quad (5)$$

$$\mathbf{u}^*(t) = \mathbf{S}^\mathsf{T}\mathbf{x}^*(t) \quad (6)$$

It can be seen to have the appropriate equilibrium $\mathbf{u}^*(\infty) = \tilde{\mathbf{S}}^{-1}\mathbf{s}^*(n)$. Note that stability is assured by the positive definiteness of $\mathbf{G}+\mathbf{S}\mathbf{S}^\mathsf{T}$.

Now, this is exactly the dynamic system which is obtained by the resistive grid shown in Figure 2. The connections are all local and of the same form as the compression network. The system could also be implemented as a standard CNN with connection radius $2r_s + 1$.

3 Halftone Compression and Quantization

The nonlinear operation performed by the H-CNN is *dynamic quantization* by which the output of each neuron becomes a binary value. This operation can be formulated in the same manner as *halftoning* [5] [7]. We would like to choose a binary image $\mathbf{y}$ which when reconstructed by a filter $\mathbf{Q}$ looks like the original image $\mathbf{s}$ filtered by $\mathbf{R}$. We will assume that $\mathbf{Q}$ and $\mathbf{R}$ are local and space-invariant. It is convenient to use the least squares criteria as a measure of how much one image looks like another.

We will use the CNN to attempt to solve for a binary output $\mathbf{y}$ to minimize the generalized least-square distortion

$$\text{dist}(\mathbf{y},\mathbf{s}) = (\mathbf{Qy}-\mathbf{Rs})^{\mathsf{T}}(\mathbf{Qy}-\mathbf{Rs}) \tag{7}$$

The desired result is to have

$$\mathbf{Qy} \approx \mathbf{Rs}$$

Let $\mathbf{s}$ be the input to a CNN. Now, the CNN was shown in [1] to always decrease the Lyapunov-energy function

$$E(t) = -\frac{1}{2}\mathbf{y}^{\mathsf{T}}(\mathbf{A}-\mathbf{I})\mathbf{y} - \mathbf{y}^{\mathsf{T}}\mathbf{Bs} \tag{8}$$

Let $\mathbf{A}$ be the matrix representation of the space-invariant template which can be written in the form

$$\mathbf{A} = -\mathbf{Q}^{\mathsf{T}}\mathbf{Q} + \text{diag}(\mathbf{Q}^{\mathsf{T}}\mathbf{Q}) + (1+\epsilon)\mathbf{I}$$

along with

$$\mathbf{B} = \mathbf{Q}^{\mathsf{T}}\mathbf{R}$$

Let $\delta = (\mathbf{Q}^{\mathsf{T}}\mathbf{Q})_{i,i}$ the value along the diagonal.

The Lyapunov-energy function becomes

$$E(t) = -\frac{1}{2}\mathbf{y}^{\mathsf{T}}(-\mathbf{Q}^{\mathsf{T}}\mathbf{Q} + (\delta+\epsilon)\mathbf{I})\mathbf{y} - \mathbf{y}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{Rs}$$

$$E(t) = \frac{1}{2}\mathbf{y}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{Qy} - \mathbf{y}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{Rs} - \frac{1}{2}(\delta+\epsilon)\mathbf{y}^{\mathsf{T}}\mathbf{y} \tag{9}$$

The $-\frac{1}{2}(\delta+\epsilon)\mathbf{y}^{\mathsf{T}}\mathbf{y}$ term is the norm-increasing term which ensures that the steady state will be in the saturation region giving a binary output. Note that this term is a constant for all steady state outputs, so there is some reason to believe that it has little effect.

The first two terms are equivalent to minimizing the distortion as defined in Equation 7 since

$$\text{dist}(\mathbf{y},\mathbf{s}) = (\mathbf{Qy}-\mathbf{Rs})^{\mathsf{T}}(\mathbf{Qy}-\mathbf{Rs}) = \mathbf{y}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{Qy} - \mathbf{y}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{Rs} - \mathbf{s}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\mathbf{Qy} + \mathbf{s}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\mathbf{Rs}$$

The input $\mathbf{s}$ is a constant vector so minimizing $\text{dist}(\mathbf{y},\mathbf{s})$ is equivalent to

$$\min_{\mathbf{y}\in\{-1,1\}^{MN}} \frac{1}{2}\mathbf{y}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{Qy} - \frac{1}{2}\mathbf{y}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{Rs} - \frac{1}{2}\mathbf{s}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\mathbf{Qy} = \frac{1}{2}\mathbf{y}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{Qy} - \mathbf{s}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\mathbf{Qy} \tag{10}$$

So, the CNN is nearly decreasing the chosen distance function during dynamic operation. Now, since $\mathbf{Qy} \approx \mathbf{Rs}$ in the least squares sense we can write

$$\mathbf{s} \approx \mathbf{R}^{-1}\mathbf{Qy}$$

so that we can approximately recover $\mathbf{s}$ from $\mathbf{y}$ by a linear filtering operation which can be implemented by a D-CNN. This is what was desired, so reconstruction is performed by

$$\mathbf{s}^* = \mathbf{R}^{-1}\mathbf{Qy}$$

As a special case, $\mathbf{R}$ can be chosen such that $\mathbf{B} = \mathbf{I}$ and therefore only the A-template is required for coding. Then $\mathbf{B} = \mathbf{Q}^{\mathsf{T}}\mathbf{R} = \mathbf{I}$ so $\mathbf{R}^{-1} = \mathbf{Q}^{\mathsf{T}}$. Therefore the reconstruction can be done by

$$\mathbf{s}^* = \mathbf{Q}^{\mathsf{T}}\mathbf{Qy} \tag{11}$$

which has impulse response the same diameter as the A-template (in fact, $\mathbf{Q}^{\mathsf{T}}\mathbf{Q}$ is the negative of the A-template with the diagonal restored) and can then be implemented as single feedforward template in the D-CNN.

As a result, the B-template is not required for halftoning and the A-template and B-template synthesis problems can be discussed *separately* for information coding and decoding.

4 Putting it all Together

We will now show how the compression and reconstruction systems just described can be combined into the CNN framework. From the block diagram and the discussion in Sections 2 and 3, it can be seen that the transmission of image n is described by:

$$\mathbf{u}_C(n) = \mathbf{u}(n) - \mathbf{u}^*(n-1)$$

$$\text{C-CNN} \qquad \begin{aligned} \tfrac{d}{dt}\mathbf{x}(t) &= -\mathbf{x}(t) + \mathbf{A}\mathbf{y}(t) + \mathbf{S}\mathbf{u}_C(n) \\ \mathbf{y}(t) &= f(\mathbf{x}(t)) \end{aligned}$$

$$\text{ideal channel} \quad \mathbf{y}^*(n) = \mathbf{y}(\infty)$$

$$\text{D-CNN} \qquad \left\{ \begin{aligned} \tfrac{d}{dt}\mathbf{x}^*(t) &= -\mathbf{G}\mathbf{x}^*(t) + (-\mathbf{S})\mathbf{u}_D(t) + \mathbf{Q}^{\mathsf{T}}\mathbf{Q}\mathbf{y}^*(n) \\ \mathbf{u}_D(t) &= \mathbf{S}^{\mathsf{T}} g(\mathbf{x}^*(t)) \end{aligned} \right.$$

$$\mathbf{u}^*(n) = \mathbf{u}_D(\infty) + \mathbf{u}^*(n-1)$$

The relation to the standard CNN is clear. The C-CNN fits the model except that $\mathbf{S}$ is not square – which is a minor issue. The D-CNN has a matrix multiplication at the output which does not match the CNN model. Note that this is not a problem for the state equation since the product $(-\mathbf{S})\mathbf{S}^{\mathsf{T}}$ represents a standard A-template. Also, it may be necessary to scale the inputs to the CNN so that they fall in the assumed range.

The compression acheivable by this method is $\frac{r}{l} \times \frac{1}{b}$ where b represents the standard number of bits used per pixel (often $b = 8$).

The function g was chosen to be linear for our experiments. However, there are many possibilities for the use of a nonlinear g. For example, it may be useful to guarantee that the ouput is in the proper range or to enhance edges.

5 Simulation Results

A computer simulation was developed to validate and demonstrate our results. A discrete-time simulation of the differential equations was used to simulate the C-CNN. The steady-state of the D-CNN was arrived at through a relaxation algorithm.

For the structural compression, the square grid shown in Figure 2 was used with unity weighting. The reconstruction was done with a self conductance of $G = 1/64$ and capacitance $C = 1$.

For halftoning, the 5 × 5 sampled Gaussian was chosen for the A-template. The choice is based on the intuition that the importance of errors to the viewer of adjacent cells matters less with distance. The template values used are given by

$$\mathbf{T}_{A_C} = \begin{pmatrix} 0.0000 & -0.0104 & -0.0208 & -0.0104 & 0.0000 \\ -0.0104 & -0.0625 & -0.1042 & -0.0625 & -0.0104 \\ -0.0208 & -0.1042 & 1.0500 & -0.1042 & -0.0208 \\ -0.0104 & -0.0625 & -0.1042 & -0.0625 & -0.0104 \\ 0.0000 & -0.0104 & -0.0208 & -0.0104 & 0.0000 \end{pmatrix}$$

and

$$\mathbf{T}_{A_D} = \begin{pmatrix} 0.0000 & 0.0104 & 0.0208 & 0.0104 & 0.0000 \\ 0.0104 & 0.0625 & 0.1042 & 0.0625 & 0.0104 \\ 0.0208 & 0.1042 & 0.1667 & 0.1042 & 0.0208 \\ 0.0104 & 0.0625 & 0.1042 & 0.0625 & 0.0104 \\ 0.0000 & 0.0104 & 0.0208 & 0.0104 & 0.0000 \end{pmatrix}$$

The system was used to encode and decode successive images of a woman in motion. The results of applying the CNN coding and decoding are shown in Figure 3. For the experimental system, the compression rate achieved was $\frac{1}{16}$.

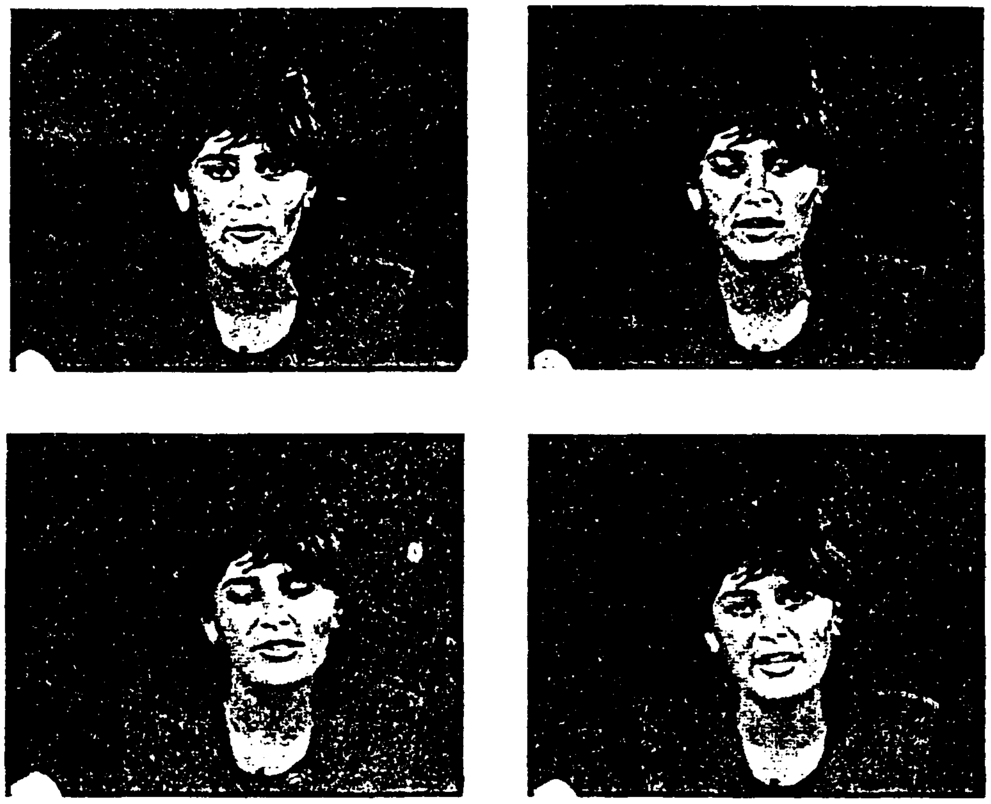


Figure 3: The 15th original image and the (50th, 100th, 149th) regenerated images.

Acknowledgments

The authors wish to express their appreciation to Prof. L. O. Chua for his suggestions and to Mr. M. Ikegami and Mr. Y. Nakamura for their efforts of making CNN programs.

References

[1] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory," *IEEE Transactions on Circuits and Systems*, vol. 32, Oct. 1988.

[2] L. O. Chua and L. Yang, "Cellular Neural Networks: Applications," *IEEE Transactions on Circuits and Systems*, vol. 32, Oct. 1988.

[3] T. Roska and L. O. Chua, "Cellular Neural Networks with nonlinear and delay-type template elements," in *IEEE International Workshop on Cellular Neural Networks and Their Applications, Proceedings*, pp. 12–25, 1990.

[4] T. Roska and L. O. Chua, "The dual CNN analog software," Technical Report DNS-2-1992, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, 1992.

[5] K. R. Crounse, T. Roska, and L. O. Chua, "Image halftoning with Cellular Neural Networks," Memorandum UCB/ERL M91/106, University of California at Berkeley, Nov. 1991.

[6] M. Tanaka, Y. Nakamura, M. Ikegami, K. Kanda, T. Hattori, Y. Chigusa, , and H. Mizutani, "Image compression and regeneration by nonlinear associative silicon retina," *Transactions of Electronics Information and Communication Engineering (Japan)*, vol. E75-A, pp. 586–594, May 1992.

[7] S. Kollias and D. Anastassiou, "A unified neural network approach to digital image halftoning," *IEEE Transactions on Signal Processing*, vol. 39, pp. 980–984, Apr. 1991.

CNN BASED ON MULTI-VALUED NEURON AS A MODEL OF ASSOCIATIVE MEMORY FOR GREY-SCALE IMAGES

Naum N. Aizenberg[+], Igor N. Aizenberg[++]

[+] University of Uzhgorod, professor of the Department of Cybernetics;
Geroev Stalingrada 28, kv.49, Uzhgorod, 249015, Ukraine
Phone: (+7 03122) 23908, Fax: (+7 03122) 36120

[++] Joint Venture PGD, Research Department Chief, Dr.Sc.;
Engelsa 27, kv.32, Uzhgorod, 294015, Ukraine
Phone: (+7 03122) 63269, Fax: (+7 03122) 36120

This work is supported by Maltian company DKL Ltd -
198 Old Bakery Street, Valetta, Malta. Fax: (+356) 221893

Abstract

In this paper we consider mathematical model of multi-valued neural element which is suggested to be used as one of the alternatives of the basic CNN element. On the basis of such CNN we offer the model of the associative memory for storing grey-scale images. Efficient learning algorithm for neural element and associative memory is presented in the paper.

I. INTRODUCTION

CNN, introduced in [1] and having been extensively developed recently [2 and others], became brilliant alternative to conventional computers for image processing and recognition.

Specifically, on the basis of CNN were introduced a number of models of associative memory and were developed the methods of solving the problems of image recognition and filtering [2 - 4]. Despite the fact that a lot of complicated tasks are solved on the basis of CNN, still we assume that in a great number of papers the range of processed signals is "unfairly" restricted - either mostly binary (bipolar) or exclusively binary signals (in case of associative memory designing) are considered. To break this restriction and extend both CNN functionality and the range of problems that are solved on CNN basis, we suggest using neural element based on the mathematical model of multi-valued threshold element [5] as a basic CNN neuron, as well as an alternative to neurons considered in [1, 6]. Proceeding from the idea that application of this element is natural for multi-valued signals processing and considering the quickly convergent learning algorithm (presented below), we believe that application of this neuron can be highly efficient. For instance, designing cellular associative memory for storing grey-scale images and implementation of image filtering algorithms on such CNN is absolutely natural. Another advantage of this basic CNN neuron is convenience of its analog (including optical) hardware implementation, since operations with complex numbers provide "arithmetical basis" of the neuron.

II. MULTI-VALUED CNN BASIC ELEMENT

To simplify the analysis, we will consider discrete time version of the CNN, though it is evident that all the results presented below are easily trasformed to the continuous time case.

Suppose we have CNN of a dimension N·M. We will apply *multi-valued neural elements* as basic neurons of this network. Each of these elements performs the following transformation:

$$Y_{ij}(t+1) = F\,[\,W_0 + \sum_m W_m^{ij}\,X_m^{ij}(t)\,], \qquad (1)$$

where Y_{ij} - neuron state, W_m^{ij}, X_m^{ij}- connection weights corresponding to m-th input of ij-th neuron and input signal value on m-th input of ij-th neuron respectively. $F(\cdot)$ - output function of ij-th neuron that will be defined furtheron.

Let input signals X and output signal Y for each neuron be located in the range 0, ..., k-1, i.e. each neuron at each particular moment performs some function of k-valued logic determined by weights W. It is evident that for byte images k = 256.

Let's assume that each value of 0, ..., k-1 corresponds to complex number r_j, j = 0, 1, ..., k-1 like this:

$$r_j = exp(i \cdot 2\pi \cdot j/k) \qquad (2)$$

where i is an imagenary unit. E.g., $r_0 = 1$, $r_1 = \varepsilon$, primitive k-th power root of a unit, $r_2 = \varepsilon^2$, ..., $r_{k-1} = \varepsilon^{k-1}$. Following this input and output signals for each neuron will be coded by k-th power roots of a unit.

Let $arg(Z)$ - argument of complex number Z ($0 \leqslant arg(Z) < 2\pi$). Now we can determine output function F for neuron (refer to (1)). Let's define function $CSIGN(Z)$, $Z \subset C$ (C-Complex Numbers Field) as follows:

$$\mathbf{CSIGN}(z) = exp(i \cdot 2\pi \cdot j/k) = \varepsilon , \qquad \text{if } 2\pi \cdot j/k \leqslant arg(Z) < 2\pi(j+1)/k,$$
$$\mathbf{CSIGN}(0) = \varepsilon^0 = 1. \qquad (3)$$

The interpretation of the (3) may be the next. If complex plane is separated into k equal sectors and complex number Z is located in j-th sector then function CSIGN(Z) equals ε^j (j = 0, 1, ..., k-1; ε is a primitive k-th power root of a unit).

Specifically CSIGN function will be used as output function F of the neuron. In this case (1) will be transformed to:

$$Y_{ij}(t+1) = \mathbf{CSIGN} [W_0 + \sum_m W_m^{ij} X_m^{ij}(t)]. \qquad (4)$$

So, (4) describes dynamics of the ij-th neuron. Naturally, since Y and X are complex numbers then weights W are complex too. In specific case when k = 2 and weights are real, (4) will describe linear threshold element and according to (3) CSIGN will become natural function SIGN.

III. LEARNING ALGORITHM FOR MULTI-VALUED NEURON

Further we will describe learning algorithm for multi-valued neural element. In the case k = 2 the problem is reduced to the well-known problem of Rosenblatt's Perceptron learning. Here we will consider the case when $k \geqslant 2$, paying special attention to the case $k > 2$.

Let $k \geqslant 2$ be an arbitrary natural number. Let's have k of non-intersecting learning subsets $A_j = \{X_1^j, ..., X_N^j\}$; j = 0, 1, ..., k-1, $X = (X_0, X_1, ..., X_n)$, where $X_0 = 1$, and other coordinates take their values from the set $\{\varepsilon^0, \varepsilon^1, ..., \varepsilon^{k-1}\}$. When $A_0, \ldots, A_{k-1}$ are predetermined, the task of learning consists in figuring out of permutation $(\alpha_0, \alpha_1, ..., \alpha_{k-1})$ and vector $W = (W_0, W_1, ..., W_n)$ satisfying

$$\text{CSIGN}(X, \bar{W}) = \varepsilon^{\alpha_j} \qquad (5)$$

for each $X \in A_j$, j = 0, 1, ..., k-1, where $\bar{W} = (\bar{w}_0, \bar{w}_1, ..., \bar{w}_n)$ - vector complex-conjugated to W; $(X,\bar{W}) = w_0 + w_1x_1 + ... + w_nx_n$ - scalar product of (n+1)-dimensional complex-valued vectors in (n+1)-dimensional space. If (5) is true, sets $A_0, A_1, ..., A_{k-1}$ are called *edge-separable*.

Since we consider neural network, we can assume that permutation $(\alpha_0, \alpha_1,$

..., α_{k-1}) is known, because it will be determined by the state of neurons from nearest neighbourhood of given neuron.

Now we have to make iterative evaluation of sequence S_w of weight vectors W_0, W_1, ..., for it to satisfy condition (5) starting from some number m and $W_m = W_{m+1} = \dots$.

The sequence S_w is evaluated as follows:

$$W_{m+1} = W_m + \omega_{s_m} \cdot D_m \cdot \varepsilon^{\alpha_j} \cdot \bar{X}_m , \qquad (6)$$

where $D_m > 0$ - correction coefficient, s_m is equal 0, 1, 2 or 3 depending on one of the following cases:

a). $\omega_0 = 0$, if $\qquad \mathrm{CSIGN}(X_m, \bar{W}_m) = \varepsilon^{\alpha_j}$;

b). $\omega_1 = -i\varepsilon$, if $\qquad \mathrm{CSIGN}(X_m, \bar{W}_m) = \varepsilon^{\alpha_j+1}$ for k = 2, 3 and

$$\varepsilon^{\alpha_j+1} \prec \mathrm{CSIGN}(X_m, \bar{W}_m) \prec \varepsilon^{\alpha_j+[k/4]} \quad \text{for } k \geqslant 4;$$

c). $\omega_2 = 1$, if

$$\varepsilon^{\alpha_j+[k/4]+1} \prec \mathrm{CSIGN}(X_m, \bar{W}_m) \prec \varepsilon^{\alpha_j+3[k/4]-1} \quad \text{for } k \geqslant 4;$$

d). $\omega_3 = i$, if

$$\mathrm{CSIGN}(X_m, \bar{W}_m) = \varepsilon^{\alpha_j+2} \quad \text{for } k = 3 \text{ and}$$

$$\varepsilon^{\alpha_j+3[k/4]} \prec \mathrm{CSIGN}(X_m, \bar{W}_m) \prec \varepsilon^{\alpha_j+k-1} \quad \text{for } k \geqslant 4.$$

Here [k/4] is an integer part of k/4, " $\prec$ " is defined as follows:

$$\varepsilon^p \prec \varepsilon^q \iff p \leqslant q \pmod{k}.$$

Let's clarify the meaning of rule (6). Note that in our case $D_m = D = 1$, m = 0, 1, ..., because all the components of all input vectors X have a constant modulo (magnitude) equal to 1. In case a) vector $W = W_m$ satisfies (5) when $X = X_m$, so there is no reason to change it. In cases b) - d) weight vector is corrected by component-by-component addition to W_m of some vector depending in each case of the difference between $\varepsilon^{-\alpha_j}\mathrm{CSIGN}(X_m, \bar{W}_m)$ and ε^0. The second addendum in (6) is evaluated from following reason:

$$(X_m, \bar{W}_{m+1}) = (X_m, \bar{W}_m) + (X_m, \omega_{s_m} \cdot \varepsilon^{\alpha_j} \cdot X_m) =$$

$$= (X_m , \bar{W}_m) + \omega_{s_m} \cdot \varepsilon^{\alpha_j} (X_m, X_m) = (X_m , \bar{W}_m) + (n+1)\omega_{s_m} \cdot \varepsilon^{\alpha_j} ,$$

where n is number of neuron inputs. ω_{s_m} is introduced to make the value of expression

$$\mathrm{CSIGN}(X_m, \bar{W}_{m+1}) = \mathrm{CSIGN}\ [(X_m, \bar{W}_m) + (n+1)\omega_{s_m} \cdot \varepsilon^{\alpha_j}]$$

closer to ε^{α_j} at each next iteration.

Initial value W_0 for vector W can be arbitrary (furtheron we will consider its selection for associative memory learning).

So iterative learning process is reduced to sequentional application of formula (6) taking into account rules a) - d) till case a) is reached. It is evident that if α_j is fixed, this process is converged maximum on step 5-6. If permutation $(\alpha_0, \alpha_1,..., \alpha_{k-1})$ is fixed, then $W_0^{\alpha_0}$ is the first to be evaluated according to (6). Then we use the obtained vector for evaluation of $W_0^{\alpha_1}$, etc. , every time checking new weight vector for previous α_j. This procedure goes on until vector W_m satisfies (5).

THEOREM 1. (On convergence of learning algorithm). If learning subsets A_0, A_1, ..., A_{k-1} are edge-separable, then it will take finite number of steps to obtain vector W_m satisfying (5) for all learning subsets.

If we presume that learning process is infinite then we will obtain contradiction with edge-separation of learning subsets.

So investigation of the possibility of neuron learning in each particular case is reduced to the investigation of the edge-separation of learning subsets. Note if this issue is settled positevely, iterative learning process is converged pretty fast. The number of iterations can reach hundreds, sometimes - thousands, but not more.

IV. MODEL OF ASSOCIATIVE MEMORY ON THE BASE OF CNN FROM MULTI-VALUED NEURONS

For solving the task of storing m patterns (grey-scale images) determined by matrixes P^1, P^2, ..., P^m of $N * M$ dimension and each image contains k of grey-scales: 0, 1, ..., k-1, let's use CNN based on multi-valued neurons. The task is reduced to evaluation of such weights W that any of P^i patterns could be reconstructed during finite number of steps from the input pattern $\tilde{P}^i$ that differs from P^i in a number of points.

So let's recode given patterns P from integer-valued to complex-valued $(0 \longrightarrow \varepsilon^0, \ldots, (k-1) \longrightarrow \varepsilon^{(k-1)})$. Then for evaluation of the connected weights between ij-th and pq-th neurons we will use the following Hebb rule [3] generalization for complex-valued patterns:

$$W^0_{ij,pq} = \begin{cases} 0 & \text{for } |i-p| > 1 \text{ or } |j-q| > 1 \\ \sum_{l=1}^{m} \bar{P}^l_{ij} P^l_{pq} & \text{for } |i-p| \leqslant 1 \text{ or } |j-q| \leqslant 1 \end{cases}$$

where P^l_{ij} is an ij-th point of l-th input pattern, " $\bar{\ }$ " is the complex-conjugation.

Having evaluated all W^0 weights for all the neurons, we obtain initial value for weights of our associative memory. Further, for evaluation of final values of weights W that will determine dynamics of the associative memory, it is necessary to apply learning algorithm based on (6) for each neuron from CNN. It is evident that the speed of learning algorithm operation will be the higher the more patterns P are correlated among themselves or , which is the same in our case, the less is the Euclid distance between patterns.

References

[1] L.O. Chua and L. Yang, "Cellular neural networks: Theory", IEEE Trans. Circuits Syst., Vol. 35, pp. 1257-1290, Oct. 1988

[2] "Proceedings of the 1990 IEEE International Workshop on CNN and their applications (CNNA-90)", Budapest, 1990, IEEE Catalog No. 90TH0312-9.

[3] S.Tan, J.Hao and J.Vandevalle, "Cellular Neural Networks as a Model of Associative Memories", in the same Proceedings, pp. 26-35.

[4] G.G.Yang, "Optical Associative Memory with Invariances", in the same Proceedings, pp. 291-301.

[5] N.N.Aizenberg, Y.L.Ivaskiv, "Multi-valued Threshold Logic", Naukova Dumka Publisher House, Kiev, 1977 (in Russian).

[6] L.O.Chua, L.Yang, K.R.Krieg, "Signal Processing using Cellular Neural Networks", Journal of VLSI Signal Processing, Vol. 3, No. 1/2 1991, pp. 25-51.

Parameter Tolerances and Generalisation Abilities of Cellular Neural Networks

Olga Kufudaki, Mirko Novák

Institute of Computer and Information Science, Prague
18207 Prague 8, Pod vodáranskou věží 2, Czechoslovakia
Phone: (00422) 821639, Fax: (00422) 8585789
E-mail: CS 15 @ cspgcs 11.BITNET

Abstract:

In this contribution the problem of cellular neural network parameter tolerances is discussed with special regard to the network generalization abilities.

1. Introduction

In the contemporary state of art of the theory of artificial neural networks in general and also of cellular neural networks especially a good part of attention is given to the problems related to efficient methods of network learning and to the questions concerning the neural network approximation abilities. The people dealing with the cellular neural network realization give a lot of attention to finding of structures, which can be implemented to VLSI chips and slices.

Very few of these authors have payed their interest to one very fundamental problem, which is the influence of designed artificial neural network parameters deviations from their theoretically optimal (nominal) values, found by the use of theoretical procedures of network synthesis, on the overall cellular network function.

2. Tolerance Problems

In the general system theory, already many years the problem is known, how to optimize the nominal system design so that the influence of these parameter changes will be minimized. The optimization is usually made so that the fundamental (primary) system properties (e.g. their transformation functions) are conserved while some of the secondary properties, e.g. the parameter tolerances are optimized. In the space of parameter tolerances this means usually, that such tolerance polyhedra is searched, which involves the maximal amount of acceptable system realizations. The solution of this problem is easy in the trivial cases only, where the straightforward methods of system analysis are applicable. However, if the number of system parameters is slightly higher (several few tenths), such approach is to be replaced by the considerably complicated numerical procedures based usually on statistical methods. The respective computation is often complicated and time consuming and therefore a lot of human mental energy was given to the search for effective and efficient design optimization methods.

The solution of the tolerance optimization problem for neural networks in general is not solved up to now. Nevertheless, some attempts in this respect were published elsewhere (see [1,2]).

When dealing with cellular artificial neural networks, we face to a little better situation. While the complexity of the mentioned problem is straightly related to the number of optimized parameters, we can use the advantage of the local connections existing between individual neuronal cells. Therefore, the wave matrices, with which we have to deal are much more spare. Their optimization with respect to the influence of parameter deviations is solvable with much less effort and also with much less powerful computer tools. For this some of the methods, developed on the base of statistical derivatives and design centering are applicable.

In cellular neural networks however, some other problems appear:

- one of them is the optimization of the structure as whole, i.e. the solution of the optimal connection between the eventually nonuniform connected neural cells or their nuclei;
- second problem consists in the fact, that the sensitivity of network properties, i.e. of the network transformation function depends on the number of network parameters (i.e. weights in this case) and decreases with the redundancy of the structure in general. Therefore, when dealing with cellular neural networks one can expect higher sensitivity with respect to cell parameters.

Also this kind of problems are principally solvable through design centering. The design procedure can be applied here in two ways:

1. on the unlearned structures as a tool for final fine network training, where it often helps to reach lower error values than obtainable by the use of standard learning procedures only (of course, according our experience, the starting from some pre-learned point in the respective multidimensional space is of the advantage),
2. on the satisfactory trained structures as a tool for finding such solution, which gives better properties of the secondary category, e.g. with respect to generalization ability.

Generalization in general represents the ability of the network to give the expected response also on inputs coming from regions, which were not trained. If the term "expected" is reasonable defined, one of the main questions is to find the minimal training regions, which are satisfactory for to reach the requested output in other regions (see [3] e.g.). The analysis of parameter tolerances allows us to attack this question from another side, i.e. to try to find the value of influence of weight matrices for which the changes of investigated secondary network properties do not change more than the prescribed ε value.

From such an analysis for individual cells we can made some estimation on also for the whole cellular neural structure (e.g. through expansion in series of nonlinear functions in the set of given points in the training regions). In the case of cellular neural networks the expected accuracy of such an estimation can be considerably good, because the respective nonlinear transformation is applied only once (for one layer network) and the mathematical expressions of the expanded series are related on sparse weight matrices only.

References

[1] Novák M., Šebesta V.: Optimization of neural network training by design centering. Neural Network World, **2**, No.2, 1992, 191-202.

[2] Novák M.: Some Considerations on the Tolerances and Sensitivities of Artificial Neural Networks. Res.Rep.No.V-495, Institute of Computer and Information Science, Prague, 1991.

[3] Kufudaki O., Vítková G., Húsek D.: Some generalization Problems of Neural Nets Res.Rep.No. V-501, Institute of Computer and Information Science, Prague, 1992

Optical Signal Processing for CNN's

Ernst Lueder Norbert Fruehauf

Institut für Netzwerk u. Systemtheorie, Universität Stuttgart
Pfaffenwaldring 47, D-7000 Stuttgart 80 , Germany
Tel.: +49-711-685-7332; Fax: +49-711-685-7311

Abstract

Signal processing in Cellular Neural Networks (CNNs) consists of multiplying and adding of signals, of calculating correlations and of shaping signals with a nonlinear function. These operations are transformed into optical processing steps by using spatial light modulators (SLMs) and lenses as basic optical elements. The SLM performs multiplications massively in parallel and with the speed of light whereas lenses realize a Fourier transform. A special system with two SLMs and two lenses is able to represent an array of CNNs without severe limitations on the interconnections between neurons. The paper gives an introduction and an overview on optical signal processing for CNNs. It is concluded by an example for optical pattern recognition.

1 Introduction

Analog and digital signal processing performed electronically is a well established art as all necessary components are available. For all practical cases they are even available in miniaturized or integrated form. Therefore other physical means of processing such as optical ones should only be considered if they offer convincing advantages. One is inclined to go optical for the enhancement of speed of a single operation since it may be performed very fast with the speed of light. However, as we shall see later, for many applications this may not be the overriding advantage of optical processing, especially because one has to consider the time for a sequence of operations. We'll investigate the issue for Cellular Neural Networks (CNN's) introduced by L. Chua [1] [2]. After recalling the basic equations and circuits for Chua's CNN's we shall have a comparative look at the electronic and optical processing required. This is followed by a more detailed description of the components and systems for optical signal processing. The presentation will be concluded by an example for optical CNN's.

2 The Cellular Neural Network

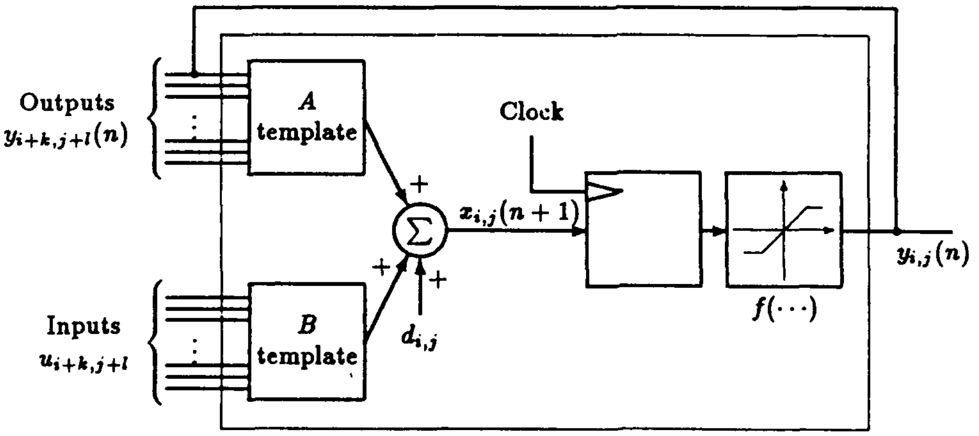


Figure 1: Architecture of neuron $C(i,j)$

In electronic realisations CNNs are as a rule only connected with their direct neighbors. An interaction with more neighbors is desirable. If N neurons were fully interconnected we had for each neuron $N-1$ incoming lines from other neurons in addition to one line from its own feedback resulting in N^2 interconnections among all neurons. Even for moderate numbers of N the interconnections would be no more feasible on an IC even not with several layers of metalization.

The signal processing within a neuron with the architecture in figure 1 is described by the state equation

$$x_{i,j}(n+1) = \sum_{k=-r}^{r}\sum_{l=-r}^{r} a_{k,l}y_{i+k,j+l}(n) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} b_{k,l}u_{i+k,j+l} + d_{i,j} \tag{1}$$

which represents analog sampled data processing at the discrete times $n, n+1$ etc. . It could also be formulated for digital processing. The terms in equation (1) are:

$x_{i,j}(n+1)$	the state of neuron i, j at the time slot $n+1$
$a_{k,l}$	the feedback operator (A–template)
$y_{i,j}(n)$	the output of neuron i, j at the time slot n
$b_{k,l}$	the control operator (B–template)
$u_{i,j}$	the input at neuron i, j
$d_{i,j}$	a bias which shifts the nonlinear function
r	the size of the neighborhood

Reciprocal time-discrete CNN's with arbitrary sigmoid functions f converge to their proper output [3]:

$$y_{i,j}(n) = f(x_{i,j}(n)) \tag{2}$$

The signal processing required for CNNs is as follows:

i) multiplications of the signals $y_{i,j}$ and $u_{i,j}$ with the weights $a_{k,l}$ and $b_{k,l}$ in the templates for which operational amplifiers with a network of resistors can be used; in digital realizations expensive multipliers are necessary. The multiplications should preferably be carried out in parallel yielding a large number of multipliers.

ii) Equation (1) contains two crosscorrelations between the weights in the templates and the signals for which again adders are needed. Electronically this again is performed by operational amplifiers with resistors for the adding. Adding comes relatively easy for analog or digital circuits if it is done electronically.

iii) The three terms in equation (1) have to be added.

iv) The nonlinearity f again is realizable by an operational amplifier or by a network of biased diodes. The shape of f can be formed relatively free and easy by electronic means.

v) In digital CNN's delay must be inserted in the feedback path in order to avoid nonrealizable delay-free loops.

vi) For a large size of r of the neighborhood the interconnections would lead to a large number of metallization-layers on an IC. Practical limit presently are, however, three layers.

Next we have to look at properties of optical components.

3 The spatial light modulator (SLM)

The liquid crystal (LC) flat panel display known from TV-systems is one of the basic elements for optical processing. Firstly its operation is outlined before we turn to the specifics of optical processing. The LC-cells in figures 2a) and 2b) contain the LC-material usually of the twisted nematic type embedded between two orientation layers on two glass plates. Between the two transparent electrodes underneath the orientation layers a voltage U can be applied. If no voltage is applied as in figure 2a) the lengthy molecules of the LC orient themselves in a helix where the groves of the orientation layers hold the molecules on each side in a given position oriented perpendicular to each other. Because of the helix the material is called twisted nematic. Incoming light is linearly polarized by a polariser-foil. The structured arrangement of the LC-molecules provides an electrical and optical anisotropy of the LC-material with speed of light depending on the direction of the light. This can be explained by a different speed of the ordinary and the extraordinary beam. The effect is called birefringence and is the cause for a rotation of the polarized light by 90^0 if the thickness d of the LC-material is chosen properly. The analyzing polarizer is oriented in the same direction as the incoming polariser. Thus polarized light turned by 90^0 cannot pass the second analyzer. The cell appears dark. Passing of light is blocked. If a sufficiently large voltage U is applied as in figure 2b), the LC-molecules orient themselves in parallel to the electrical field. In this mode no rotation of the polarization occurs and

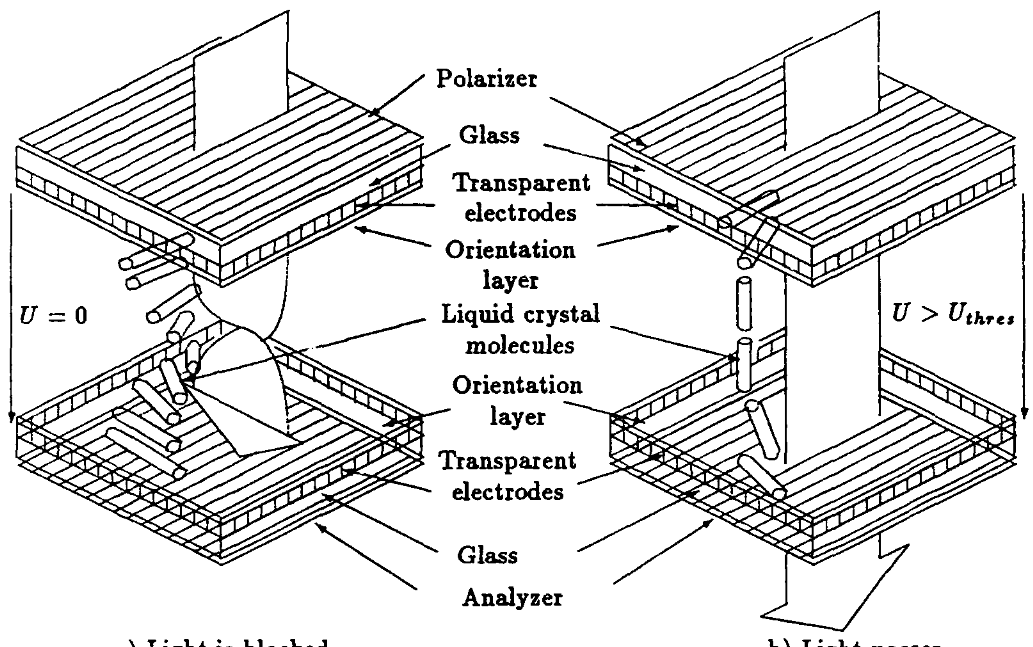


Figure 2: The principle of operation of a TN cell

the incoming light fully passes through the analyzing polarizer. The cell appears bright. For lower voltages U a rotation of the polarization by less than 90^0 is observed and a component of the light can pass through the second polarizer. The cell appears grey. The grey shade is controlled by the level of the voltage U applied. The surface of the glass plate is subdivided into an array of rectangular picture elements (pixels) each of which can be individually controlled by a voltage. This array of pixels is called a spatial light modulator (SLM).

The transparency T of a pixel for incoming light is a complex function as not only the amplitude but also the phase of the light wave is affected by the LC-material. $|T|$ and arc$T$ are given as [4]:

$$|T| = \frac{1}{2\pi} \cdot \frac{1}{\frac{1}{4} + \frac{d^2}{\lambda^2}\left(n_e(U) - n_o\right)^2} \cdot \sin \pi^2 \left[\frac{1}{4} + \frac{d^2}{\lambda^2}\left(n_e(U) - n_o\right)^2\right] \qquad (3)$$

$$\text{arc}\, T = \frac{\pi \cdot d}{\lambda}\left(n_e(U) - n_o\right) \pm m \cdot \pi \qquad ;m = 0, 1, 2, \cdots \qquad (4)$$

with

n_o	is the refractive index of the ordinary beam
$n_e(U)$	is the voltage dependent refractive index of the extraordinary beam

The nonlinear function $|T(U)|^2$ is shown in figure 3. This nonlinearity is used for CNN's. The addressing of the pixels can be achieved electronically or optically. The circuit for the electronic addressing is drawn in figure 4a). The thin film transistor (TFT) operates as a switch. If the TFT's in a line of the SLM are turned conductive by a positive gate impuls the video-signal determining the grey shades is fed in through the column lines and charges the LC-capacitance C_{LC} and the additional thin film storage capacitance C_p in a pixel. This is simultaneously done to all pixels in a line of the display. The information is fed in one line at a time. After charging the capacitances within the time t_L the TFT's are turned off by which the charge is maintained on the $C's$. The rotation of the LC-molecules is not yet completed after t_L. However, since the voltage across the $C's$ is maintained the rotation is still allowed to continue to its final position. After t_L the next line can be addressed. The time t_P required to write in a full picture in an SLM with n lines is $t_P = n\ t_L$ reflecting in a picture frequency $f_P = 1/t_P = 1/(n\ t_L)$. The time t_P is essential for the programming of an SLM with new picture information.

For a typical cell with an twisted nematic (TN) LC-material and fast TFT's with CdSe as semiconductor $t_L = 5\mu s$ [5]. Thus a picture with $n = 1000$ lines yields $t_P = 5ms$ and $f_P = 200Hz$. This is

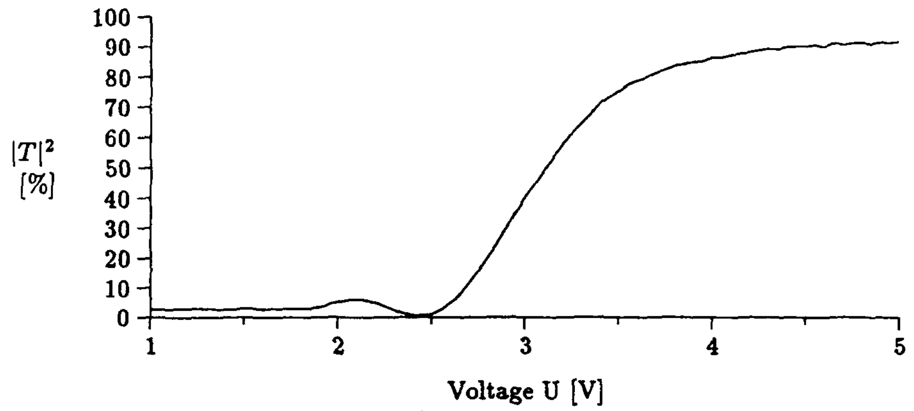


Figure 3: Transmission–versus–voltage characteristic of a TN–cell

the largest picture frequency presently available with TN- cells. The time t_L represents the speed bottleneck in optical processing. Till now electronically addressed SLMs with up to $575 \times 2088 = 1200600$ picture elements each with a size of $130 \times 48\mu m^2$ have been produced in our laboratory [6].

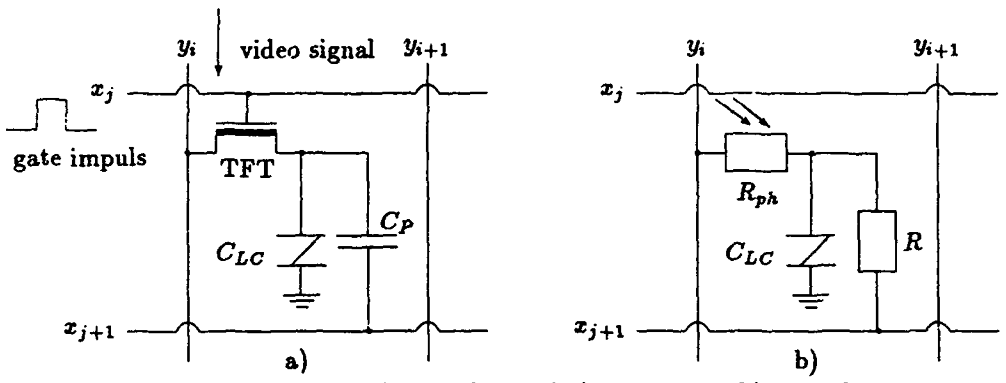


Figure 4: Addressing circuit of a pixel a) electronical b) optical

The optical addressing of an LC-cell is depicted in figure 4b). An incident light beam hits a photoconductive layer and lowers its resistance R_{ph} due to the intensity of the light. This changes a resistive voltage-divider such that the voltage across a pixel of the LC-cell increases yielding a higher transmissivity of the LC-material. The cell in figure 4b) has to be operated in a reflective mode since the photoconductor is not transparent. The light to be modulated is shone onto the cell from the same side as the reflected beam emits.

4 Optical components of signal processing

The components are treated in the same sequence as in paragraph 2.

4.1 The optical multiplication

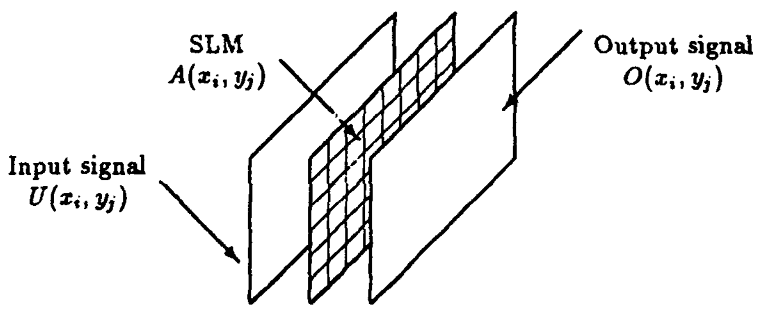


Figure 5: Parallel and analog optical multiplication

The liquid crystal display or an SLM in figure 5 exhibits a 2-dimensional array of pixels at the location x_i, y_j each of which has an electronically or optically controlled transmission. We assign a pixel to each neuron. If we place as input signals a picture $U(x_i, y_j)$ with given grey shades $u_{i,j}$ in the N pixels x_i, y_j behind the SLM $A(x_i, y_j)$ in figure 5 with also N pixels exhibiting the grey shades $a_{i,j}$ and shine light through both then the outgoing light intensity represents the analog multiplication [7]

$$O(x_i, y_j) = U(x_i, y_j)A(x_i, y_j) \tag{5}$$

as each light beam through a pixel x_i, y_j is attenuated by the equation

$$o_{i,j} = u_{i,j}a_{i,j} \tag{6}$$

SLM's with $N \geq 10^6$ are presently fabricated. Therefore the set-up in figure 5 realizes more than a million analog multiplications in parallel and with the speed of light. This massive and fast parallelism is one of the biggest assets of optical processing.

4.2 The optical crosscorrelation

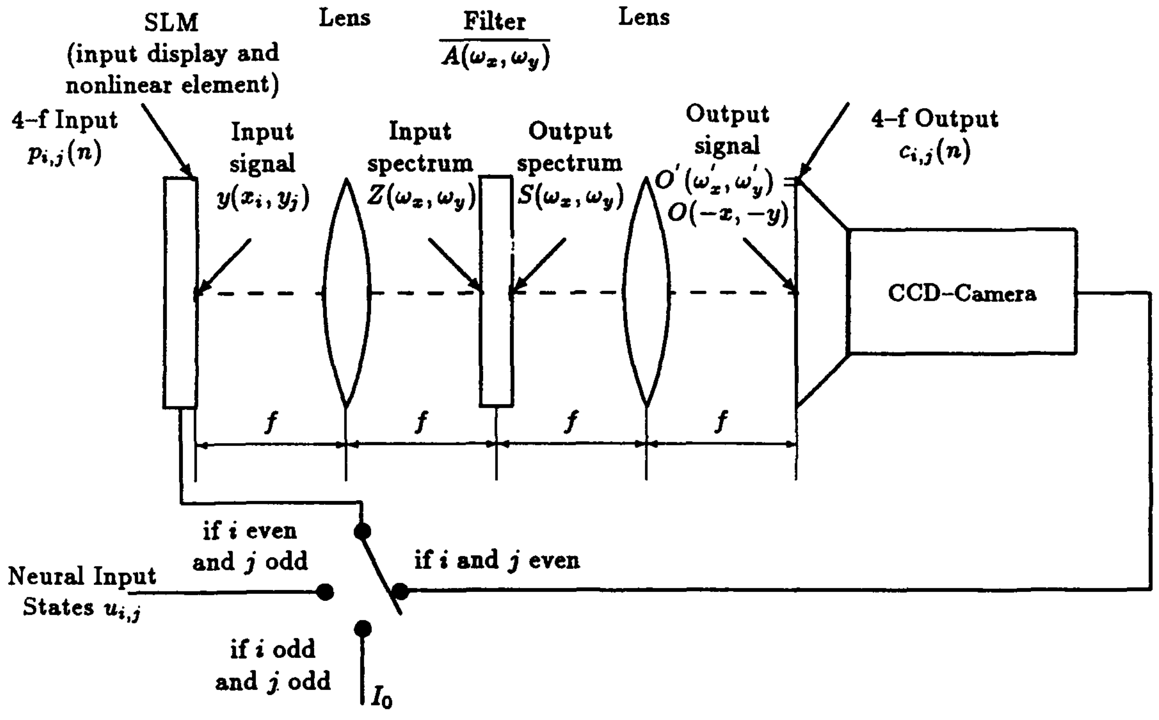


Figure 6: An optical Time-Discrete CNN

It is known in optics [7] that the first lens in figure 6 with focal length f performs a 2-dimensional Fourier transform of a picture or a field in the front focal plane with the complex light wave amplitude $Y(x_i, y_j)$. The Fourier transform is the spectrum

$$Z(\omega_x, \omega_y) = \frac{j}{\lambda \cdot f} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} Y(x_i, y_i) \exp\left(-j(\omega_x x_i + \omega_y y_i)\right) dx_i\, dy_i \tag{7}$$

with the spatial frequencies in x- and in y-direction

$$\omega_x = \frac{2 \cdot \pi}{\lambda \cdot f} x_o \quad \text{and} \quad \omega_y = \frac{2 \cdot \pi}{\lambda \cdot f} y_o$$

as well as

j	the imaginary unit $j = \sqrt{-1}$
λ	the wavelength of the used light wave

The scaling factor in front of equation (7) is independent of the variables and is therefore from now on neglected. The equation (7) is an approximation due to Fresnel which only holds for thin lenses and paraxial light rays.

Now we turn to a 4-f system which consists of the two lenses in figure 6. The spectrum $Z(\omega_x, \omega_y)$ in equation (7) is located in the back focal plane of the first lens where it is multiplied by $\overline{A(\omega_x, \omega_y)}$ using an SLM. The bar stands for the conjugate complex of A. The spectrum

$$S(\omega_x, \omega_y) = Z(\omega_x, \omega_y)\, \overline{A(\omega_x, \omega_y)} \tag{8}$$

at the output of the SLM is again Fourier-transformed by the second lens yielding

$$O'(\omega'_x, \omega'_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} S(\omega_x, \omega_y) \exp\left(-j(\omega_x\omega'_x + \omega_y\omega'_y)\right) d\omega_x\, d\omega_y \tag{9}$$

We now have to clarify the meaning of $O'(\omega'_x, \omega'_y)$ in the new plane of spatial frequencies ω'_x and ω'_y. An inverse Fourier-transform is not available with lenses. Since we are accustomed to the result of the inverse Fourier-transform leading from the domain of spatial frequencies back to the space-plane we look at the space-plane expression corresponding to $S(\omega_x, \omega_y)$, namely

$$O(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} S(\omega_x, \omega_y) \exp\left(j(\omega_x x + \omega_y y)\right) d\omega_x\, d\omega_y \tag{10}$$

A comparison of equations (9) and (10) reveals that the result O' obtained by the second lens coincides with O by interpreting the variables ω'_x and ω'_y as

$$\omega'_x = -x \qquad \text{and} \qquad \omega'_y = -y \tag{11}$$

yielding

$$O'(-x, -y) = O(x, y) \tag{12}$$

Therefore the mirror-image of O' is identical to the inverse Fourier-transform of $S(\omega_x, \omega_y)$ in equation (8). On the other hand the inverse Fourier transform in equation (10) can also be calculated by a convolution of the inverse Fourier transforms $a_0(x, y)$ of $\overline{A(\omega_x, \omega_y)}$ and $y(x, y)$ of $Z(\omega_x, \omega_y)$, where $a_0(x, y)$ is the point spread function of $\overline{A}$ corresponding to the impulse response of a transfer function.
The convolution yields

$$O(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a_0(\alpha, \beta) y(x - \alpha, y - \beta) d\alpha\, d\beta \tag{13}$$

With the substitution $-\alpha = \alpha'$ and $-\beta = \beta'$ we obtain

$$O(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a_0(-\alpha', -\beta') y(x + \alpha', y + \beta') d\alpha'\, d\beta' \tag{14}$$

If $a(x, y)$ as the spatial function to the spectrum $A(\omega_x, \omega_y)$ is real for real x and y, then a well known theorem states that $\overline{A(\omega_x, \omega_y)}$ has the inverse transform $a_o(x, y) = a(-x, -y)$. Thus equation (14) yields now again with variables α and β

$$O(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a(\alpha, \beta) y(x + \alpha, y + \beta) d\alpha\, d\beta \tag{15}$$

This is the crosscorrelation between $a(x, y)$ and $y(x, y)$. The result in equation (15) can be phrased as follows: The 4-f system in figure 6 performs a crosscorrelation between $y(x, y)$ and $a(x, y)$ if $y(x, y)$ represents the input field and the conjugate complex of the spectrum of $a(x, y)$ is used in the first focal plane of the second lens. However, the output $O(x, y)$ is the mirror image of the desired crosscorrelation $O'(x, y) = O(-x, -y)$.

SLM's exhibit a discretisation of the x-y plane into pixels x_i, y_j which for briefty are denoted by i, j. The correlation integral in equation (15) becomes the double sum

$$O(i,j) = \sum_{\alpha=-\infty}^{\infty} \sum_{\beta=-\infty}^{\infty} a(\alpha,\beta) y(i+\alpha, j+\beta) \tag{16}$$

or with denotations closer to equation (1)

$$O(i,j) = \sum_{k=-r}^{r} \sum_{l=-r}^{r} a_{k,l}\, y_{i+k,j+l} \tag{17}$$

where the finite extension r of the summing has been taken into account. This limitation is possible because $a_{k,l} \neq 0$ only for $k = -r, -r+1, \cdots, r-1, r$ and $l = -r, -r+1, \cdots, r-1, r$. The system in equation (17) has $(2r-1)^2$ free parameters. The operation in equations (15), (16) and (17) are called space invariant because the output O depends on the difference rather than on the actual value of the arguments of $a(x,y)$ and $y(x,y)$. The correlation equation (17) realizes the desired correlations in equation (1) by massive and fast parallel processing.

The crosscorrelator has a physical length of $4f$, therefore it is usually designated as a "4-f-system". It is possible to reduce the physical length of the optical correlator to about $1.4f$ if the simple lenses are replaced by lens systems [8].

4.3 The optical addition

The three terms in equation (1) have to be added. In strong contrast to electronics adders are hard to achieve in optics. One could add light intensities with photosensitive resistors which decrease its resistance with increasing light intensity. However, this effect is highly nonlinear; in addition the characteristics of the resistors fluctuate widely from sample to sample which makes high precision signal processing virtually impossible. Therefore we implement additions by making use of the additions associated with the crosscorrelation in equation (17) by putting all three terms in equation (1) into one single matrix [9] [10]. We combine the inputs to the templates in figure 1 into one variable $p_{i,j}(n)$ in equation (18) as further detailed in figure 7.

$$p_{i,j}(n) = \begin{cases} y_{i/2,j/2}(n) & \text{i even, j even} \\ u_{i/2,(j+1)/2} & \text{i even, j odd} \\ I_0 & \text{i odd} \end{cases} \tag{18}$$

i \ j	1	2	3	4	$\cdots$
1	I_0	I_0	I_0	I_0	$\cdots$
2	$u_{1,1}$	$y_{1,1}$	$u_{1,2}$	$y_{1,2}$	$\cdots$
3	I_0	I_0	I_0	I_0	$\cdots$
4	$u_{2,1}$	$y_{2,1}$	$u_{2,2}$	$y_{2,2}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Figure 7: Combined Input coding scheme for $p_{i,j}$

Similarly we combine the A- and B- templates and the bias $d_{i,j}$, which is used to shift the nonlinearity f, according to equation (19) and figure 8

$$t_{k,l} = \begin{cases} a_{k/2,l/2} & \text{k even, l even} \\ b_{k/2,(l+1)/2} & \text{k even, l odd} \\ I_{k+1/2,l} & \text{k odd} \end{cases} \tag{19}$$

The bias function $I_{k,l}$ is chosen according to:

$$\sum_{k=-r}^{r} \sum_{l=-2r-1}^{2r} I_{k,l} I_0 = d_{i,j} \tag{20}$$

$t_{k,l}$ is the kernel of the optical correlator. The optically computed correlation $c_{ij}(n)$ between $p_{i,j}$ and $t_{k,l}$ contains the state variable $x_{ij}(n+1)$ as the elements with even i, j. Thus

$$\begin{aligned} c_{2i,2j}(n) &= x_{i,j}(n+1) = \sum_{k=-2r-1}^{2r} \sum_{l=-2r-1}^{2r} t_{k,l} p_{2i+k,2j+l}(n) = \sum_{k=-r}^{r} \sum_{l=-r}^{r} t_{2k,2l} p_{2i+2k,2j+2l}(n) + \\ & \sum_{k=-r}^{r} \sum_{l=-r}^{r} t_{2k,2l-1} p_{2i+2k,2j+2l-1}(n) + \sum_{k=-r}^{r} \sum_{l=-2r-1}^{2r} t_{2k-1,l} p_{2i+2k-1,2j+l}(n) \end{aligned} \tag{21}$$

A look at the meaning of the indices of $t_{i,j}$ and p_{ij} in equations (19) and (18) reveals

	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$\cdots$	$I_{-1,-3}$	$I_{-1,-2}$	$I_{-1,-1}$	$I_{-1,0}$	$I_{-1,1}$	$I_{-1,2}$	$\cdots$
$\cdots$	$b_{-1,-1}$	$a_{-1,-1}$	$b_{-1,0}$	$a_{-1,0}$	$b_{-1,1}$	$a_{-1,1}$	$\cdots$
$\cdots$	$I_{0,-3}$	$I_{0,-2}$	$I_{0,-1}$	$I_{0,0}$	$I_{0,1}$	$I_{0,2}$	$\cdots$
$\cdots$	$b_{0,-1}$	$a_{0,-1}$	$b_{0,0}$	$a_{0,0}$	$b_{0,1}$	$a_{0,1}$	$\cdots$
$\cdots$	$I_{1,-3}$	$I_{1,-2}$	$I_{1,-1}$	$I_{1,0}$	$I_{1,1}$	$I_{1,2}$	$\cdots$
$\cdots$	$b_{1,-1}$	$a_{1,-1}$	$b_{1,0}$	$a_{1,0}$	$b_{1,1}$	$a_{1,1}$	$\cdots$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Figure 8: Combined Template coding scheme

$$x_{i,j}(n+1) = \sum_{k=-r}^{r}\sum_{l=-r}^{r} a_{k,l}y_{i+k,j+l}(n) + \sum_{k=-r}^{r}\sum_{l=-r}^{r} b_{k,l}u_{i+k,j+l} + \sum_{k=-r}^{r}\sum_{l=-2r-1}^{2r} I_{k,l}I_0 \qquad (22)$$

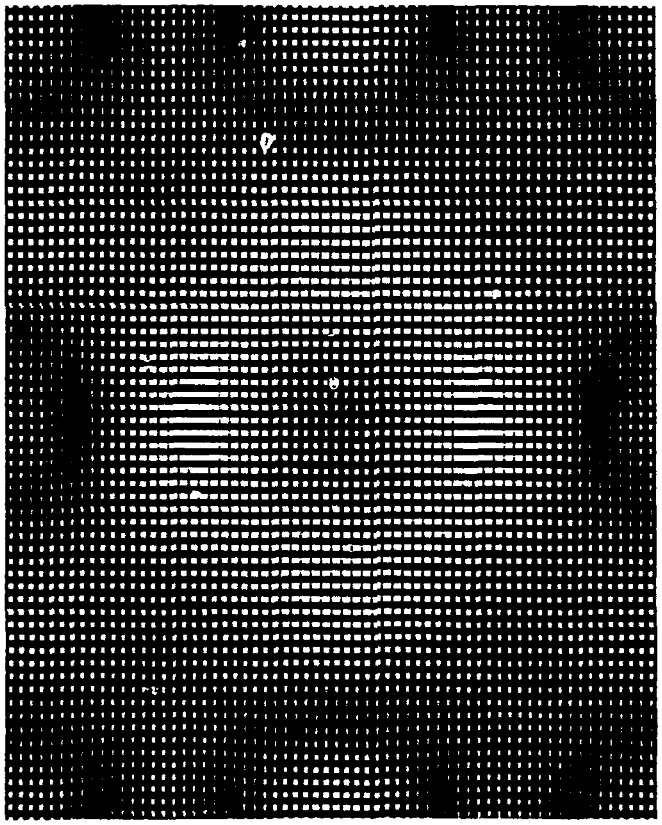


Figure 9: Hologram for a corner detector (enlarged)

The detection of the correlation result $c_{2i,2j}(n)$ poses another problem because all sensors (i.e. CCD cameras or photoresistors) are sensitive only to the intensity of the impinging light wave. As the intensity is proportional to the squared magnitude of the complex light wave amplitude, it is impossible to distinguish between positive and negative values of $c_{2i,2j}(n)$. This ambiguity can be removed if a constant $D = |\min c_{2i,2j}(n)| = |\min x_{i,j}(n+1)|$ is added before the sensing. In the actual optical realization this will be done by adding D to the right side of equation 20:

$$\sum_{k=-r}^{r}\sum_{l=-2r-1}^{2r} I_{k,l}\cdot I_0 = d_{i,j} + D \qquad (23)$$

The increase of the state variable $x_{i,j}(n+1)$ by a constant D may be canceled with a shift of the nonlinear output function $f(x)$ in equation 2:

$$f\left(x_{i,j}(n+1)\right) = \tilde{f}\left(x_{i,j}(n+1) + D\right) \qquad \text{where} \qquad \tilde{f}(x) = f(x - D) \tag{24}$$

The nonlinearity will be described in the next paragraph.

4.4 The optical nonlinearity

The nonlinearity is realised as outlined in paragraph 3 by the nonlinear transmission of the SLM. The optical system representing a CNN is shown in figure 6.

4.5 The optical storage

Storage for photons is not so readily feasible as storing of electrons on a capacitor. However, new LC-materials permit to overcome this shortcoming of optical systems. Ferroelectric LC-materials exhibit two stable positions of its molecules, one in which the cell is blocked and a second one in which it is transparent. The molecules assume these positions with an appropriate addressing impuls and stay there also after the electronic addressing has been removed [11]. This effect allows for the storage of images or of a bit-pattern as required in digital signal processing.

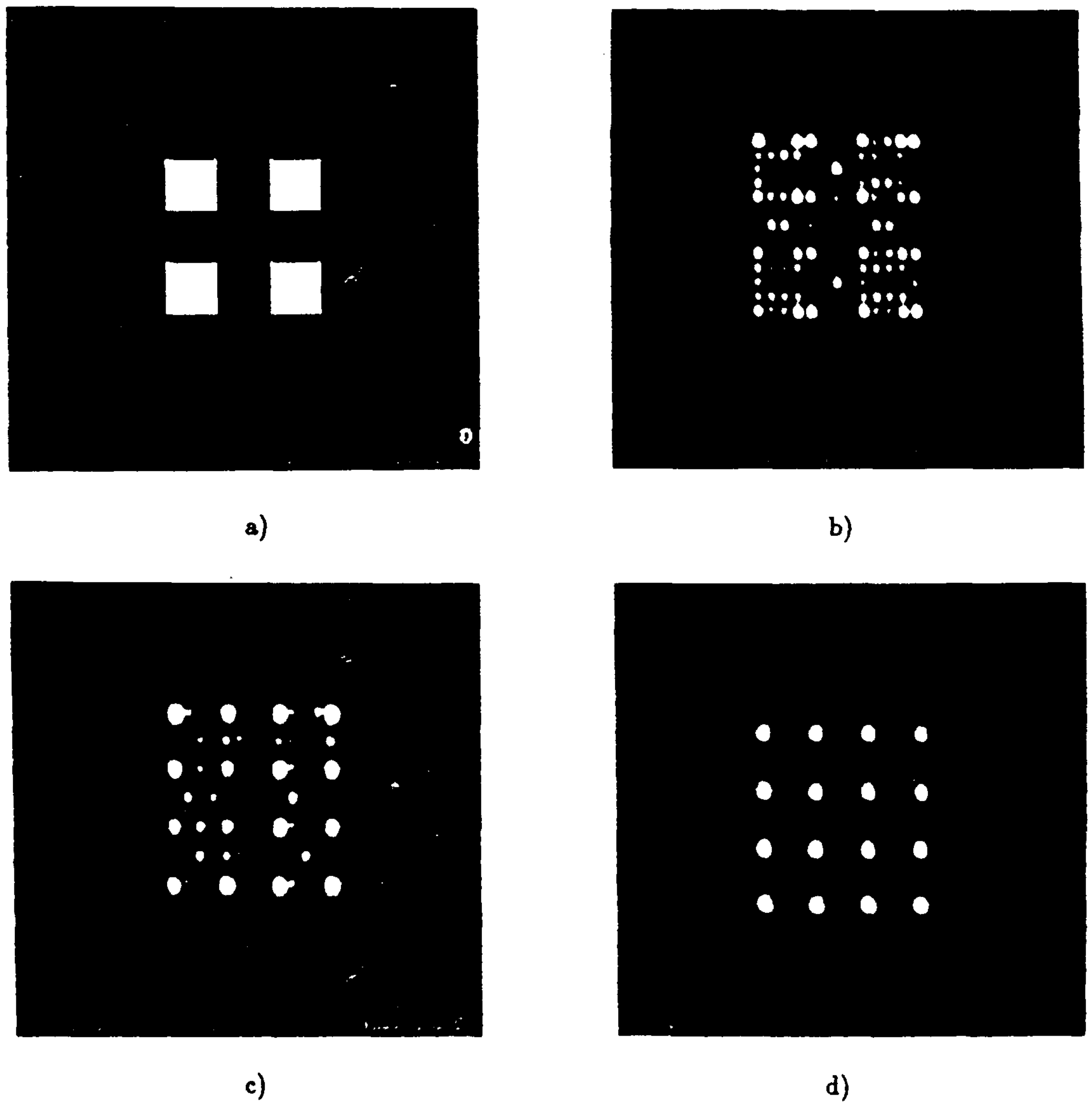


Figure 10: Corner Detection of four squares

4.6 The interconnections

The most attractive feature of optical processing is the fact that the interconnections between the neurons are provided inherently and in a massive parallel form by the Fourier transform of the lens. Thus the problems encountered with interconnecting metallization layers on a chip just do not exist in optical processing. The weights of the interconnections are choosen according to the combined template $t_{k,l}$ in equation (19). The conjugate spectrum of $t_{k,l}$ in the filter plane of the crosscorrelator can be realized by an SLM or as in the following experiment by a computer generated hologram. The hologram is a binary amplitude filter which is based on a Lohmann coding scheme with circular overflow [12] [13]. In this coding scheme, the area and the location of small holes in an opaque screen are choosen according to the magnitude and the phase of the required complex valued filter. Figure 9 depicts a filter which realizes the combined template for a corner detector. An example of an optical CNN which performs corner detection will be given in the following paragraph.

5 Operation of the optical CNN

Figure 10 shows an example of a CNN which performs a corner detection of four squares. The original input picture is depicted in figure 10 a). Figures 10 b) - 10 d) show the combined output in time slot 1, 2 and 4 during iteration thereby confirming the correct operation of the optical CNN.

References

[1] L.O. Chua and L. Yang, "Cellular neural networks: Theory", *IEEE Trans. Circuits Syst.*, **vol. 35**, pp. 1257 - 1272

[2] L.O. Chua and L. Yang, "Cellular neural networks: Applications", *IEEE Trans. Circuits Syst.*, **vol. 35**, pp. 1273 - 1290

[3] N. Fruehauf, L.O. Chua, E. Lueder, "Convergence of reciprocal Time-Discrete Cellular Neural Networks with continuous nonlinearities", *Proc. Second International Workshop on Cellular Neural Networks and Applications, München*, 1992

[4] Kanghua Lu, B.E.A. Saleh, "Theory and design of the liquid crystal TV as an optical spatial phase modulator", *Opt. Engineering*, **vol. 29**, pp. 240-246, Mar. 1990

[5] M. Dobler, E. Lueder, H.-U. Lauer, T. Kallfass, "Frame sequential Projection Displays with High Speed and High Voltage TFTs", *2nd International Cadmium Selenide Workshop, Imperial College London*, 1992

[6] V. Hochholzer, E. Lueder, H.U. Lauer, E. Ginter, D. Straub, T. Kallfass, J. Minarsch, "A MIM-Addressed 1.2-MPEL Light Valve for a Rear-Projector Display", *Soc. Inform. Display, Int. Symp., Digest of Techn. Papers*, pp. 51 – 54 ,1992

[7] J.W. Goodman, "Introduction to Fourier optics", *Mc Graw-Hill Book Co.*, New York 1968

[8] B.A. Blandford, "A New Lens System for use in Optical Data-Processing", in *Conf. Optical Instruments and Techniques, Reading 1969*, pp. 435 – 443

[9] N. Fruehauf, E. Lueder, "Realization of CNNs by optical parallel processing with spatial light valves", in *Proc. First IEEE Int. Workshop Cellular Neural Networks Appl., CNNA-90 (Budapest)*, 1990, pp. 281 - 291

[10] N. Fruehauf, E. Lueder, M. Gaida, G. Bader, "An optical implementation of space invariant Cellular Neural Networks", in *Proc. 10'th European Conf. on Circuit Theory and Design, ECCTD-91*, 1991, pp. 42 - 51

[11] H. Rieger, C. E. Escher, G. Illian, H. Jahn, A. Kaltbeitzel, T. Harada, A. Weippert, E. Lueder, "FLCD Showing High Contrast and High Luminance", *Soc. Inform. Display, Int. Symp., Digest of Techn. Papers*, pp. 396 – 399, 1991

[12] D. Schreier, "Synthetische Holografie", *Physik Verlag*, 1984

[13] W.H. Lee, "Computer-Generated Holograms: Techniques and Applications", *Prog. Opt.*, **vol. 16**, pp. 119-232, 1978

Robustness of attractor networks and an improved convex corner detector

P. Nachbar[*], A.J. Schuler, T. Füssl, J. A. Nossek[†], L.O. Chua[‡]

Abstract

By defining several notions of robustness for an attractor network, we are able to augment previous results about the AdaTron algorithm by explicit values for the robustness of the optimal weights. We show, that the symmetry of a problem is reflected by the invariance of the optimal weights. This enables us to deduce that a convex corner detection, using a discrete-time Cellular Neural Network (DTCNN), can not be accomplished with just one clock cycle, and we propose an improved convex corner detector.

1 Introduction

It was shown, that the AdaTron (*Ad*aptative Percep*tron*) algorithm is able to find the most robust weights of a perceptron, just taking actual constraints into account [2, 3, 4, 11]. Algorithms solving the perceptron problem [13, 10] can be applied to the learning problem of attractor networks, provided one is able to specify the trajectories explicitly. Hence we were able to specify an algorithm which yields the optimal weights for a rotationally invariant (or isotropic [14]) r-neighborhood DTCNN [5, 12] on a square grid. By rotationally invariant we mean that any rotation of $k \cdot 90°$ with $k \in N$ around the center of the template and any reflection along the diagonals through the center, leaves the template invariant. In the first section we augment these results by providing explicit formulas for the insensitivity of the optimal solution and define several notions of robustness for an attractor network. Further we will show, if a task is invariant with respect to some group, so are the optimal weights. As a side effect it can happen, that the optimal weights are invariant with respect to a larger group of transformations. This leads to the observation that the conventional convex corner detector must fail in some situations. Using this information as a guide we propose an improved one.

2 Robustness of attractor networks

Let us consider discrete time neural networks [9, 1, 8] i.e.

$$\mathbf{x}(l+1) = \mathrm{sgn}(\mathbf{A}\mathbf{x}(l) + \mathbf{B}\mathbf{u} + \mathbf{i}) \; ; \; \mathbf{x}(0), \mathbf{i}, \mathbf{u} \in [-1,1]^n \; ; \; \mathbf{x}(l) \in \{-1,1\}^n \text{ for } l \in N \; ; \; \mathbf{A}, \mathbf{B} \in R^{n\times n} .$$

The design problem is given by triplets of patterns $(\rho^\nu, \xi^\nu, \sigma^\nu)$, where ρ^ν is the desired output (after one time step), ξ^ν is the previous output, and σ^ν is the input pattern for $\nu = 1, \cdots, P$. Using this notation the design problem takes the form $\rho_k^\nu(\mathbf{a}_k^t \xi^\nu + \mathbf{b}_k^t \sigma^\nu + \mathbf{i}_k) > 0$ for each pattern index ν and each neuron k. $\mathbf{a}_k^t$ resp. $\mathbf{b}_k^t$ is the k-th row vector of the matrix $\mathbf{A}$ resp. $\mathbf{B}$. Note that any solution can be scaled by a positive factor yielding another solution of the design problem. It is reasonable to define the most insensitive solution (also called optimal stability [7]) by the following min-max problem:

$$\max_{\mathbf{a}_k, \mathbf{b}_k, i_k} \min_\nu \rho_k^\nu(\mathbf{a}_k^t \xi^\nu + \mathbf{b}_k^t \sigma^\nu + i_k) \text{ subject to } \| \mathbf{a}_k \|^2 + \| \mathbf{b}_k \|^2 + i_k^2 = 1 \text{ for } k = 1, \cdots, n .$$

We will provide a more rigorous justification later when we introduce the notion of robustness as used in statistical design by norm body inscription [15, 11]. Note that the constraint imposed on the "weights"

[*]Supported by a grant from the Ernst von Siemens foundation.
[†]Institute for Network Theory and Circuit Design; Technical University of Munich. Germany; Tel.: 49/89/2105-8520, FAX: 49/89/2105-8504, e-mail:pena@nws.e-technik.tu-muenchen.de
[‡]Electronics Research Laboratory, College of Engineering; University of California, Berkeley;

is, due to our previous remarks, natural and moreover the problem would be ill-defined otherwise. It can be shown [7, 2, 4], that this min-max problem is equivalent to a convex quadratic programming problem, namely

$$\min \left(\| \mathbf{a}_k \|^2 + \| \mathbf{b}_k \|^2 + i_k^2\right) \text{ subject to } \rho_k^\nu(\mathbf{a}_k^t \xi^\nu + \mathbf{b}_k^t \sigma^\nu + i_k) \geq 1 \ \forall \nu \text{ and for } k = 1, \cdots, n\,,$$

which can be solved efficiently for example by the AdaTron algorithm.

It is often necessary to impose constraints on the "weights" A, B and i, for example requiring symmetric matrices or the translation invariance of the weights which yields a DTCNN [8]. This can be done by specifying a linear parametrization $\mathbf{S} : R^m \longrightarrow R^{n \times n}$ of the "weights". We will proof that even in this case the equivalence of the min-max problem and a suitably chosen convex quadratic programming problem still holds, hence providing an efficient tool for the design of the most insensitive DTCNN.

2.1 The AdaTron Theorem

In the last section we claimed the equivalence of a certain min-max problem to a quadratic optimization problem. More precisely it was proven [7, 2, 4]:

Theorem 1 *If the perceptron problem is solvable, i.e. if there is a weight vector* $\mathbf{w} \in R^n$, *such that for all patterns* $\theta^\nu \in R^n$ $(\nu = 1, \cdots, P)$, $\mathbf{w}^t\theta^\nu > 0$, *then the following is equivalent:*

- *Finding the perceptron of optimal insensitivity, i.e. the solution of*

$$\max_{\mathbf{w}} \min_{\nu} \mathbf{w}^t\theta^\nu \textit{ subject to } \| \mathbf{w} \|^2 = 1\,. \tag{1}$$

- *Solving the convex quadratic programming problem given by*

$$\min \| \mathbf{w} \|^2 \textit{ subject to } \mathbf{w}^t\theta^\nu \geq 1 \textit{ for } \nu = 1, \cdots, P\,. \tag{2}$$

The solutions $\mathbf{w}^1$ *of (1) and* $\mathbf{w}^2$ *of (2) are related by* $\mathbf{w}^1 = \lambda \mathbf{w}^2$ *for an appropriate* $\lambda > 0$.

Given $\mathbf{w}^2$ the solution of (2) the factor λ in the theorem is necessary to ensure the normalization of $\mathbf{w}^1$, resp. given $\mathbf{w}^1$, λ is used to ensure, that for at least one pattern θ^μ the equality $\mathbf{w}^{2^t}\theta^\mu = 1$ holds. Note that since the solution of a convex quadratic programming problem is unique, also the perceptron of optimal insensitivity is uniquely defined. This theorem can be easily applied to discrete time neural networks if one sets for each neuron k $\mathbf{w}^t := (\mathbf{a}_k^t, \mathbf{b}_k^t, i_k)$ and $\theta^{\nu t} := (\rho_k^\nu \xi^{\nu t}, \rho_k^\nu \sigma^{\nu t}, \rho_k^\nu)$ and solves the quadratic programming problem for each neuron. As already noted it is useful to consider problems where not all the coefficients of the weight vector w are actually independent degrees of freedom, for example one may require that the "templates" $\mathbf{a}_k$ and $\mathbf{b}_k$ are rotationally invariant, as it is sensible for some image processing task. Let $\mathbf{S} : R^m \longrightarrow R^n$ be a linear parametrization of the weight vector w, i.e. each w can be written as $\mathbf{w} = \mathbf{S}\mathbf{v}$ for some vector $\mathbf{v} \in R^m$. Such a parametrization is not unique and any non-singular transformation $\mathbf{Q} : R^m \longrightarrow R^m$ defines a equivalent parametrization $\mathbf{S}' := \mathbf{S}\mathbf{Q}$. Using this degree of freedom the original problem (1), together with the constraint imposed by the parametrization S, can be written as

$$\max_{\mathbf{v}} \min_{\nu} \mathbf{v}^t\zeta^\nu \text{ subject to } \mathbf{v}^t\mathbf{Q}^t\mathbf{S}^t\mathbf{S}\mathbf{Q}\mathbf{v} = 1\,,$$

with $\zeta^\nu := \mathbf{Q}^t\mathbf{S}^t\theta^\nu$. If we choose Q such that $\mathbf{Q}^t\mathbf{S}^t\mathbf{S}\mathbf{Q} = \mathbf{1}_m$, where $\mathbf{1}_m$ is the m dimensional identity matrix, we can apply the original theorem and obtain a convex quadratic programming problem. Since Q and $\mathbf{1}_m$ are non-singular, this can only be achieved if $\mathbf{S}^t\mathbf{S}$ is non-singular. In this case there exists a orthogonal transformation U such that $\mathbf{U}^t\mathbf{S}^t\mathbf{S}\mathbf{U} = \mathbf{D}$ is diagonal with entries $\mathbf{D}_{ii} > 0$ and the transformation Q is given by $\mathbf{Q} = \mathbf{U}\mathbf{D}^{-0.5}$. We therefore obtain the following theorem.

Theorem 2 *Let* $\mathbf{S} : R^m \longrightarrow R^n$ *be a linear map such that* $\mathbf{S}^t\mathbf{S}$ *is non-singular. If the perceptron problem with linear constraints is solvable, i.e. there is a weight vector* $\mathbf{w} \in R^n$, *such that for all patterns* $\theta^\nu \in R^n$ $(\nu = 1, \cdots, P)$, $\mathbf{w}^t\theta^\nu > 0$, *and* $\exists \mathbf{v} \in R^m$ *such that* $\mathbf{w} = \mathbf{S}\mathbf{v}$, *then the following is equivalent:*

- *Finding the perceptron of optimal insensitivity, i.e. the solution of*

$$\max_{\mathbf{w}} \min_{\nu} \mathbf{w}^t\theta^\nu \textit{ subject to } \| \mathbf{w} \|^2 = 1 \textit{ and } \exists \mathbf{v} \in R^m : \mathbf{w} = \mathbf{S}\mathbf{v}\,.$$

- *Solving the convex quadratic programming problem given by*

$$\min \| \mathbf{v} \|^2 \;\; \textit{subject to}\; \mathbf{v}^t\zeta^\nu \geq 1 \;\textit{for}\; \nu = 1, \cdots, P\,.$$

  where $\zeta^\nu := \mathbf{D}^{-0.5}\mathbf{U}^t\mathbf{S}^t\theta^\nu$ *and* $\mathbf{U}$ *is an orthogonal matrix such that* $\mathbf{U}^t(\mathbf{S}^t\mathbf{S})\mathbf{U} = \mathbf{D}$ *is a diagonal matrix.*

If $\mathbf{v}^*$ *is a solution of the second problem the solution of the initial problem* $\mathbf{w}^*$ *is given by* $\mathbf{w}^* = \lambda \cdot \mathbf{SUD}^{-0.5}\mathbf{v}^*$ *for an appropriate* $\lambda > 0$. *The value of the optimal insensitivity* $\max_{\mathbf{W}}^* \min_\nu \mathbf{w}^{*t}\theta^\nu = \lambda$.

The last assertion is easily checked using the properties of the transformations $\mathbf{U}$ and $\mathbf{D}$. The assumption that $\mathbf{S}^t\mathbf{S}$ is non-singular just reflects the fact that the parametrization $\mathbf{S}$ may not be redundant. Since if $\mathbf{S}^t\mathbf{S}$ were singular, there would be a parameter vector $\mathbf{v} \neq \mathbf{0}$ such that $(\mathbf{Sv})^t(\mathbf{Sv}) = 0$. But this is only possible if $\mathbf{Sv} = \mathbf{0}$ and therefore there would be two different parameter vectors ($\mathbf{v}$ and $\mathbf{0}$) yielding the same weight vector.

2.2 Robustness of attractor networks

In the last subsection we extended the AdaTron theorem and showed how it is applied to discrete time neural networks if no constrains are present. We introduced for each neuron site j a $2n+1$ dimensional weight vector $\mathbf{w}$. Let us now introduce the weight matrix $\mathbf{W} \in R^{n\times(2n+1)}$, where the j-th row vector (written as a column vector) is the previously introduced vector $\mathbf{w}$ for the neuron at site j, from now on named $\mathbf{w}_j$. Let us further define the vector $\tilde{\mathbf{w}} \in R^{n(2n+1)}$ by $\tilde{\mathbf{w}}^t := (\mathbf{w}_1^t, \cdots, \mathbf{w}_n^t)$. A general linear parametrization of a discrete time neural network is now given by a linear map $\mathbf{S} : R^m \longrightarrow R^{n\cdot(2n+1)}$ and we require that there is a parameter vector $\mathbf{v} \in R^m$ such that $\tilde{\mathbf{w}} = \mathbf{Sv}$. The parametrization $\mathbf{S}$ can be used for example to express the translational invariance of the weights $\mathbf{w}_j$ or the symmetry of the $\mathbf{A}$ and $\mathbf{B}$ matrix. Using this notation the learning problem is given by a pair of patterns (ρ^ν, θ^ν), where $\rho^\nu \in R^n$ and $\theta^\nu \in R^{2n+1}$ and we are looking for weights $\mathbf{W}$ such that $\mathrm{sgn}(\mathbf{W}\theta^\nu) = \rho^\nu$. This in turn is equivalent to $\rho_j^\nu \mathbf{w}_j\theta^\nu > 0$, which we will call the attractor network problem, in analogy to the perceptron problem. With this notation at hand let us now relate the notion of insensitivity to the notion of robustness as used in statistical design by norm-body inscription [6, 15]. The robustness can be defined for arbitrary norms on vector spaces. We will give the definition only for the euclidean norm, since it is the only one we will consider. For a more thorough treatment of this concept as applied to neural networks, we refer the reader to [16].

Definition: For any solution $\mathbf{W}$ of the attractor network problem (ρ^ν, θ^ν) with linear constraints $\mathbf{S}$

- the relative robustness in pattern space $r_p(\mathbf{W})$ is defined as the solution of the following optimization problem
$$\max r \;\text{ subject to }\; \forall \Delta\theta^\nu : ||\Delta\theta^\nu|| = r||\theta^\nu|| \text{ implies } \mathbf{w}_j^t\rho_j^\nu(\theta^\nu + \Delta\theta^\nu) \geq 0\,.$$
- the absolute robustness in pattern space $R_p(\mathbf{W})$ is defined as the solution of the following optimization problem
$$\max R \;\text{ subject to }\; \forall \Delta\theta^\nu : ||\Delta\theta^\nu|| = R \text{ implies } \mathbf{w}_j^t\rho_j^\nu(\theta^\nu + \Delta\theta^\nu) \geq 0\,.$$

The relative robustness in weight space $r_w(\mathbf{W})$ is defined correspondingly.

Lemma 1 *The relative robustness in pattern space and weight space are identical and can be calculated as follows*

$$r_p(\mathbf{W}) = r_w(\mathbf{W}) = \min_{\nu j} \frac{\rho_j^\nu \mathbf{w}_j^t\theta^\nu}{\| \mathbf{w}_j \| \| \theta^\nu \|}\;. \tag{3}$$

Correspondingly for R_p *follows*

$$R_p(\mathbf{W}) = \min_{\nu j} \frac{\rho_j^\nu \mathbf{w}_j^t\theta^\nu}{\| \mathbf{w}_j \|}\;.$$

Proof: Let us pick any pattern and θ^ν and any neuron j with the associated weight vector $\mathbf{w}_j$ and compute the maximal robustness just taking this pattern and neuron into account. From the geometrical representation in Fig. 1 it is obvious, that

$$\rho_j^\nu \mathbf{w}_j^t(\theta^\nu + \Delta\theta^\nu) = 0 \quad\text{and}\quad \rho_j^\nu(\mathbf{w}_j + \Delta\mathbf{w}_j)^t\theta^\nu = 0 \tag{4}$$

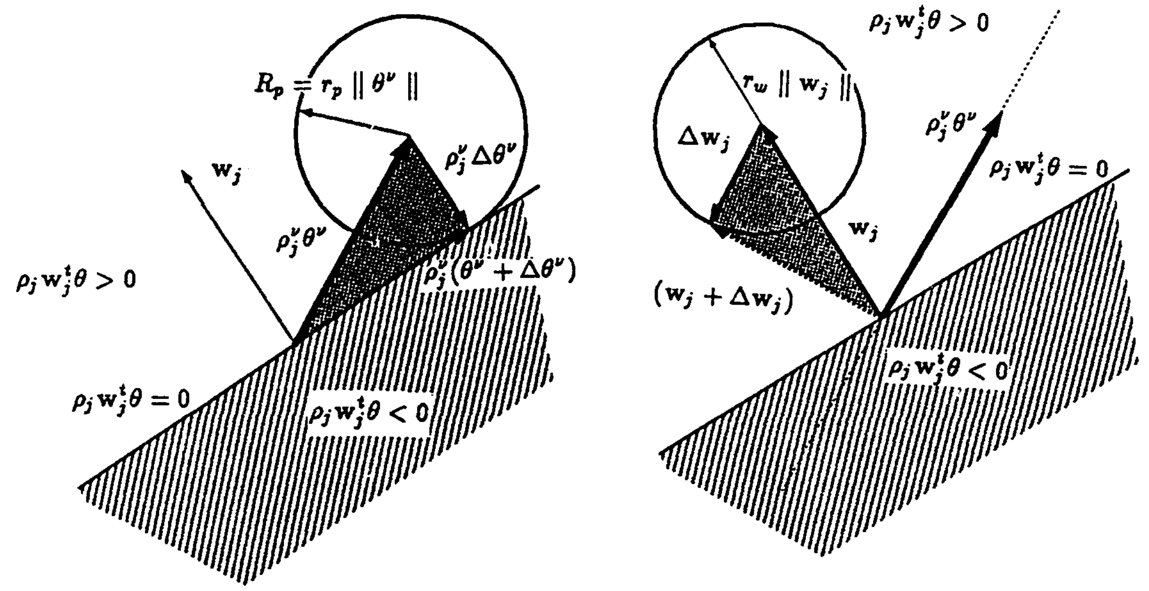

$R_p = r_p \parallel \theta^\nu \parallel$
$\mathbf{w}_j$
$\rho_j^\nu \Delta\theta^\nu$
$\rho_j^\nu \theta^\nu$
$\rho_j \mathbf{w}_j^t \theta > 0$
$\rho_j^\nu(\theta^\nu + \Delta\theta^\nu)$
$\rho_j \mathbf{w}_j^t \theta = 0$
$\rho_j \mathbf{w}_j^t \theta < 0$
$\rho_j \mathbf{w}_j^t \theta > 0$
$r_w \parallel \mathbf{w}_j \parallel$
$\Delta \mathbf{w}_j$
$\rho_j^\nu \theta^\nu$
$\rho_j \mathbf{w}_j^t \theta = 0$
$\mathbf{w}_j$
$(\mathbf{w}_j + \Delta \mathbf{w}_j)$
$\rho_j \mathbf{w}_j^t \theta < 0$

Figure 1: Representation of the hyperplane defined by $\mathbf{w}_j$ in the space of patterns θ
a) Maximum norm pertubation of a pattern
b) Maximum norm pertubation of a weight vector
still guaranteeing correct operation.

holds. For the perturbation vectors in Fig. 1a) and b) we have

$$\rho_j^\nu \Delta\theta^\nu = -\frac{\mathbf{w}_j}{\parallel \mathbf{w}_j \parallel} \cdot r_p \parallel \theta^\nu \parallel \quad , \quad \Delta \mathbf{w}_j = -\frac{\rho_j^\nu \theta^\nu}{\parallel \theta^\nu \parallel} \cdot r_w \parallel \mathbf{w}_j \parallel \ . \tag{5}$$

From (5) and (4) it is easy to show, that

$$r_p = -\frac{\rho_j^\nu \mathbf{w}_j^t \Delta\theta^\nu}{\parallel \mathbf{w}_j \parallel \parallel \theta^\nu \parallel} = \frac{\rho_j^\nu \mathbf{w}_j^t \theta^\nu}{\parallel \mathbf{w}_j \parallel \parallel \theta^\nu \parallel} = -\frac{\rho_j^\nu \Delta \mathbf{w}_j^t \theta^\nu}{\parallel \mathbf{w}_j \parallel \parallel \theta^\nu \parallel} = r_w \ . \tag{6}$$

Therefore the maximal robustness, taking all patterns into account, is nothing else, than taking the minimum over $\frac{\rho_j^\nu \mathbf{w}_j^t \theta^\nu}{\parallel \mathbf{w}_j \parallel \parallel \theta^\nu \parallel}$ for all ν and j and this completes the proof of (3). □

Remark: Depending on the specific application, either of these notions of robustness can be the appropriate one. In DTCNNs with their translationally invariant templates all weight vectors $\mathbf{w}_j$ are equal in length and we can apply the AdaTron theorem and corresponding algorithm, which can accommodate only one constraint ($\parallel \mathbf{w}_j \parallel = 1$) for designing an attractor network, which is maximally robust. Moreover, if the patterns have equal length (e.g. binary patterns) such a network will also have maximum absolute robustness in pattern space R_p. Accounting for linear constraints can be done as described in section 2.1 by applying the AdaTron algorithm to the parameter vector v and the transformed patterns ζ^ν. The maximal robustness thereby obtained in parameter and transformed pattern space, will result in maximal robustness in the original spaces provided, that the transformation S is norm preserving.

3 Convex corner detection

3.1 Group invariant learning

It is often argued, that for rotationally invariant (image processing) tasks, rotationally invariant templates are the most favorable ones. The following lemma provides a justification.

Lemma 2 *Let $\{g_1, \cdots, g_r\}$ be an orthogonal (linear) representation of a finite group G. Let the weights W be the solution of*

$$\max_{W} \min_{\nu, p} W^t(g^p \xi^\nu) \geq 0 \ \textit{subject to} \ W^2 = 1 \tag{7}$$

then the weights W are invariant with respect to this group, i.e. $g^p W = W$ for all transformations g^p.

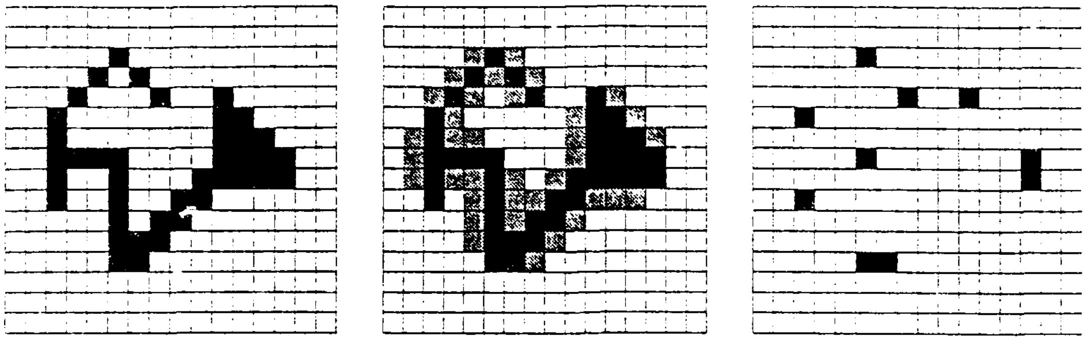


Figure 3: Training patterns

convex corners (Fig. 2). Observe, that the convex corners are characterized by the fact that the pixel of the input pattern and the initial pattern are black and that the cell in the initial pattern has at least one white vertical or horizontal neighboring cell. Finally one has to ensure, that the final output is a fixpoint. Therefore, in a third step, we tried to learn the output , together with the original input, to be a fixpoint.

For our experiments we used a 16×16 square grid and we refer to indices lower than 1 and higher than 16 as border pattern. We tried to learn the three processing steps with the initial pattern all white and all black and with the initial pattern equal to the input pattern and its inverse. In all cases it was not possible to learn all three steps. Though it is possible with an all white initial state to learn the first couple processing steps. The template values are given in Tab. 3, and can be used to perform a convex corner detection using a DTCNN which has to be stopped after two clock cycles, since otherwise it would produce an all white output.

We therefore designed templates for each one of the two processing step with the learning patterns of Fig. 2. Again we ensured, that the output is a stable fixpoint. The template values are given in Tab. 1, 2 for border pattern values equal to zero and -1 (white) and for the initial state equal to the input and a all white state (for the first processing step only). Actually the patterns in Fig. 3 were just the first ones. In order to obtain templates that are not only optimal with respect to the training patterns, but also to the task itself we have employed the following procedure. We started out with one training pattern and kept adding new patterns while monitoring the optimal template values. Since the optimal values changed by less than 0.1 percent, after 3 patterns were presented we are confident that the template values are optimal with respect to the task.

4 Conclusion

It was shown, that the AdaTron algorithm yields the optimal weights for various notions of robustness even in the presence of symmetries, just taking actual constraints into account. From an engineering point of view, the actual value of the robustness is extremely important and we had to augment our previous results. The experiments we performed for the improved convex corner detector were quite promising, since the algorithm is efficient (in our case within minutes on a Sun workstation) and the value of the robustness is fairly high. Still the main drawback of any design algorithm is, that if the problem is not solvable with just one processing step a trajectory has to be designed. For problems possessing a symmetry, we showed how "non-geometrical" transformations can hinder a solution in a single processing step. In turn this information can be used as a guide for the design of an appropriate trajectory.

References

[1] D.J. Amit, H. Gutfreund, and H. Sompolinsky. *Phys.Rev.*, A(32):1007, 1985.

[2] J.K. Anlauf and M. Biehl. *Europhys. Lett.*, (10):687, 1989.

[3] J.K. Anlauf and M. Biehl. Properties of an adaptive Perceptron algorithm. In R. Eckmiller, G. Hartmann, and G. Hauske, editors, *Parallel Processing in Neural Systems and Computers.* Elsevier, 1990.

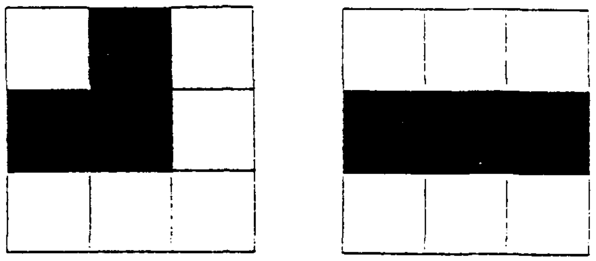


Figure 2: Patterns indistinguishable for an isotropic DTCNN

Proof: The proof is indirect. Suppose, that W is the solution of (7) and let us define the weight vector $\tilde{w}$ as $\tilde{w} := 1/r \sum_q g^q W$ and compute its length.

$$<\tilde{w},\tilde{w}> = \frac{1}{r^2}\sum_{p,q} < g^q W, g^p W > \leq \frac{1}{r^2}\sum_{p,q} ||g^q W||\,||g^p W|| = 1 \ , \qquad (8)$$

since the transformations are orthogonal. Hence there is a $\lambda \geq 1$ such that $||\lambda \tilde{w}|| = 1$ and moreover, by construction, λW is invariant with respect to the group G. For any transformation g^p and pattern ξ^ν we have:

$$(\lambda\tilde{w})^t(g^p\xi^\nu) = \frac{\lambda}{r}\sum_q W^t(g^q)^t g^p \xi^\nu = \frac{\lambda}{r}\sum_s W^t g^s \xi^\nu \geq \lambda \min_{\nu s} W^t g^s \xi^\nu \ . \qquad (9)$$

Since $\lambda \geq 1$ and the solution of (7) is unique, we have an contradiction, which proves the claim. □
The above ideas can also be applied to attractor networks. The neuron at site i of an attractor network performs the mapping $\rho_i^\nu = \text{sgn}(W_i^t\xi^\nu)$. If we require, that the patterns $g^p\xi^\nu$ are mapped in the same way, the above considerations show, that the optimal weights W_i are invariant with respect to the group G. Actually the group G could be different for each neuron, yielding different constraints for the weights. This shows that in this context we require a local invariance. As we will see for the convex corner detection it can happen that there is a larger group of transformation $\{h^1, \cdots, h^R\}$ which contains the group of transformations $\{g^1, \cdots, g^r\}$ as a subgroup, but also leaves the weights W_i invariant. It is then also impossible to distinguish between the patterns $h^p\xi^\nu$ and the patterns ξ^ν. Applying these considerations to a DTCNN it is easily checked that a isotropic (image processing) task implies that the optimal templates are isotropic ones. The coefficients of the a and b template hence satiesfy $a_{st} = a_{pq}$ iff the sets $\{|s|, |t|\}$ and $\{|p|, |q|\}$ are equal. Let us now illustrate the consequences of the above mentioned "larger group" for the learning problem. To this end we define the vector $U = (U_1, \cdots, U_4) = (a_{10}, a_{01}, a_{-10}, a_{0-1})$. A rotation about 90° is given by a cyclic permutation of the elements of U. A reflection along the diagonal permutes U_1 with U_2 and U_3 with U_4. Hence the invariance of the template implies that all 4 coefficients are equal. Such a vector is now invariant with respect to any permutation of its elements, including the "non-geometrical" permutation of U_1 with U_2. Therefore the patterns in Fig. 2 can not be distinguished.

3.2 Improved convex corner detection

Following the procedure outlined at the end of section 2.2, we derived an algorithm which yields the most robust weights for a rotationally invariant r-neighborhood DTCNN on a $(M \times N)$ square grid [11]. For the design of the templates we used this algorithm and by combining the assertion about the optimal insensitivity (theorem 2) with the definitions and explicit expressions for the robustness of attractor networks we are able to provide explicit values for the optimal robustness. The considerations at the end of the last subsection showed, that the detection of convex corners can not be accomplished using a single processing step of a DTCNN (Fig. 2). To overcome this difficulty we designed a two step process for the convex corner detection of a binary image. Since straight lines can not be distinguished from corners (Fig. 2), we use the first processing step to mark those boundary points of the image which appear to be part of a straight line by placing an additional black pixel right next to it. (gray pixels in Fig. 3). More precisely whenever a pixel is white and the pattern inside a 1-neighborhood "window" centered at this pixel appears to be part of a straight line crossing this window, we change the white center pixel into a black one. This output is employed as the initial state for the next processing step which extracts the

[4] M. Biehl, J.K Anlauf, and W. Kinzel. Perceptron learning by constrained optimization: The AdaTron algorithm. In *Neurodynamics 90*, 1990.

[5] L.O. Chua and Lin Yang. Cellular Neural Networks: Theory. *IEEE Tr. CAS*, 35(10):1257, 1988.

[6] S.W. Director and G.D. Hachtel. The simplicial approximation approach to design centering. *IEEE Tr. CAS*, (24):363, 1977.

[7] E. Gardner. *J.Phys.*, A(21):257, 1988.

[8] H. Harrer and J.A. Nossek. Discrete-Time Cellular Neural Networks. *Int. J. Circuit Theory Appl.*, July 1992. to be published.

[9] W.A. Little. *Math.Biosci.*, (19):101, 1974.

[10] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, first edition, 1969.

[11] P. Nachbar, T. Füssl, J.A.Nossek, and L.O. Chua. The AdaTron learning algorithm with constraints. Technical Report UCB/ERL M92/70, University of California, Berkeley, 1992.

[12] J.A. Nossek, G. Seiler, T. Roska, and L.O. Chua. Cellular Neural Networks: Theory and circuit design. *Int. J. Circuit Theory Appl.*, 1992. to be published.

[13] F. Rosenblatt. *Principles of Neurodynamics.* Spartan, New York, 1962.

[14] G. Seiler and J.A. Nossek. Symmetry properties of Cellular Neural Networks on a square and hexagonal grid. In *Proc. of the Second IEEE International Workshop on Cellular Neural Networks and their Application*, Munich, 1992. IEEE.

[15] G. Seiler and A.J. Schuler. Some notes on statistical design by norm-body inscription. Technical Report TUM-LNS-TR-92-2, TU Muinch, 1992.

[16] G. Seiler, A.J. Schuler, and J.A. Nossek. Design of robust Cellular Neural Networks. Technical Report TUM-LNS-TR-91-13, TU Muinch, 1991.

	border pattern value −1		border pattern value 0	
	input	all white	input	all white
a_{00}	0	4	0	4
a_{01}	0	−1.33	0	−1.33
a_{11}	0	−1.33	0	−1.33
b_{00}	6	6	12	6
b_{01}	8	8	16	8
b_{11}	4	4	8	4
I	16	14.66	30	14.66
r_w	0.06	0.06	0.03	0.06

Table 1: Optimal template values for the first processing step, for the initial pattern equal to the input and an all white pattern.

	b.p.v. −1	b.p.v. 0
a_{00}	31	31
a_{01}	−26	−26
a_{11}	−14	−14
b_{00}	31	31
b_{01}	8	8
b_{11}	4	4
I	−8	−8
r_w	0.02	0.02

Table 2: Optimal template values for the second processing step (b.p.v = border pattern value).

	b.p.v. −1	b.p.v. 0
a_{00}	8	8
a_{01}	−4	−4
a_{11}	0	0
b_{00}	6	6
b_{01}	0	0
b_{11}	0	0
I	−8	−8
r_w	0.09	0.09

Table 3: Optimal template values for the stopped process (b.p.v = border pattern value).

Boltzmann Learning of Parameters in Cellular Neural Networks

Lars Kai Hansen
CONNECT, Electronics Institute, build. 349
Technical University of Denmark, DK-2800 Lyngby, Denmark
Tlf.: +45 45 93 12 22 ext. 3889, Fax: +45 42 88 01 17
Email: lars@eiffel.ei.dth.dk

We use Bayesian methods to design cellular neural networks for signal processing tasks and the Boltzmann Machine learning rule for parameter estimation. The learning rule can be used for models with "hidden" units, or for completely unsupervised learning. The latter is exemplified by unsupervised adaptation of an image segmentation cellular network, in particular we apply the learning rule to adaptive segmentation of satellite imagery.

1 Introduction

The Bayesian or *Maximum Posterior* approach is a very successful device for signal processing [15, 7, 11], a particular attraction is that it leads to algorithms that map well onto networks of locally connected, simple processing elements ie. cellular neural networks [4]. With the advent of low-cost massively parallel hardware, this virtue may end up being decisive for (near) future real life applications. However, while the Bayesian approach allows formulation of collective models that solve complex signal processing tasks, general and flexible tools for parameter estimation are lacking. Direct maximum-likelihood estimation is hampered by the difficulty of obtaining analytical expressions for derivatives of normalization constants [1]. Besag has introduced two methods based on approximate maximum-likelihood estimation, the *coding* method [1], and the *pseudo-likelihood* method [2]. Both methods pose difficulties if the image model includes non-observable attributes ie. *hidden* units. In this contribution we discuss the use of the Boltzmann Machine learning rule [10] for parameter estimation. The learning rule may be applied to general situations with hidden units without complication. We address in this presentation *unsupervised* learning.

Since the phase-space distribution of a dynamical system in contact with a heat-bath is a Gibbs distribution; sampling from such distributions have been central to simulations of statistical physics systems. The standard simulation tool is due to Metropolis et al. [13]. Geman and Geman [7] introduced Metropolis sampling from Gibbs distributions as a simulation tool for visual reconstruction and showed that a *Simulated Annealing* strategy [12] could improve the speed of the sampling process. The sampling process implements a stochastic neural network with symmetric connections. Hinton and Sejnowski [10] studied supervised learning in such networks and introduced the term *Boltzmann Machines* to emphasize the relation to statistical physics. Invoking the *Mean Field* approximation, Peterson and Anderson derived a deterministic selfconsistent set of equations for the time-averages of the dynamical variables. Using these averages in the learning rule, they obtained substancial improvements in speed and performance [14].

0-7803-0875-1/92 3$.00 ©1992IEEE

In the next section the Bayesian approach to signal processing is outlined, and we discuss the design of cellular networks using the Mean Field approach. In section three we show how the Boltzmann Machine learning rule may be applied for parameter adaptation. Section four contains experiments on image segmentation and concluding remarks.

2 Bayesian Signal Processing

The Bayes approach to signal processing has a long tradition, see e.g. [11, 7], or [5] for a recent introduction. The basic idea is to consider both the source (un-degraded) signal and the degradation as stochastic processes. The Bayes formula can then be used to construct the distribution of the reconstructed signal (x), conditioned on the observed degraded signal (y):

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{1}$$

According to the standard interpretation the conditional distribution is the product of the distribution of the degradation process: $P(y|x) \equiv P(x \rightarrow y)$, and the *prior* distribution of the reconstructed signal $P(x)$. $P(x|y)$ of equation (1) is referred to as the *posterior* distribution. A useful estimate of the recontructed signal is given by the location of the mode of the posterior distribution, the socalled *Maximum A Posteori* (MAP) estimate. In the following we derive the posterior distributions for an image segmentation model.

2.1 Image Segmentation

Segmentation is an important step in many computer vision systems, however even in its simplest form: binarization of a grey-scale image there exist no established standard solution!. Here we use the Bayes scheme to derive a simple cost-function that can be minimized by a cellular neural network. The resulting cost-function is identical to the one used by Carnevali *et al.* [3]. The target signal is a smooth binarization of a grey-scale image d_j, in terms of two-valued pixels $S_j \in \{-1, +1\}$. The prior distribution is designed to emphasize smoothness:

$$P[S] = Z_1^{-1} \exp\left(-\sum_{j=1}^{N}\sum_{j'=1}^{N} M(j,j')(S_j - S_{j'})^2\right) \tag{2}$$

$M(j,j')$ defines the connectivity, hence the unit cell of the cellular network. Here we just use the nearest neighbors. More complex connectivity structures have been designed for modelling textural features [6].

We assume the signal degradation to consist in addition of white Gaussian noise. This degradation process leads to the following conditional distribution:

$$P[d|S] = Z_2^{-1} \exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{N}(S_j - d_j)^2\right) \tag{3}$$

We want to approach real data for which the noise variance is unknown, hence this parameter has to estimated as part of the learning process. As above we use Bayes to combine and obtain the posterior distribution. Clearly it is of the Gibbs form[1], with a cost-function given by the negative logarithm of the posterior distribution: $-\log P[S|d]$. We note that the state dependent part of the cost-function is linear in the parameters $M(j,j')$ and $w_d \equiv 1/\sigma^2$.

2.2 Network Design

The Mean Field annealing method for estimation of averages over Gibbs distributions is well documented in the litterature see e.g. Hertz *et al.* [9]. The cellular neural network is designed to minimize the Mean Field *free energy F*, and this can be done either in analog mode:

$$\tau\frac{\partial\langle S_j\rangle}{\partial t} = -\frac{\partial F}{\partial\langle S_j\rangle} = -\langle S_j\rangle + \tanh\left(\beta^t\left(\sum_{j'=1}^{N} M(j,j')\langle S_{j'}\rangle + w_d d_j\right)\right) \tag{4}$$

or in discrete time mode:

$$\langle S_j^{t+1}\rangle = (1-\frac{\Delta}{\tau})\langle S_j^t\rangle + \frac{\Delta}{\tau}\tanh\left(\beta^t\left(\sum_{j'=1}^{N} M(j,j')\langle S_{j'}^t\rangle + w_d dj\right)\right) \tag{5}$$

where the time-scale Δ/τ can be used to regularize the stability of the iteration process in digital implementation[14]. β^t is quantify the annealing schedule, in this work we use the simple schedule: $\beta^t = \beta^1 + (t/t_{max})(\beta^2 - \beta^1)$.

In summary, the unit cell of the cellular network contains one unit approximating the local thermodynamic average: $\langle S_j^t\rangle$, and one input unit d_j. We assume that $M(j,j')$ connects nearest neighbors symmetrically with weight w_n, and the weights to the input units are all w_d.

3 Boltzmann Machine Learning

In order to apply the cellular network above we have to estimate the parameters (denoted $w = (w_n, w_d)$). Since our network is based on the Gibbs distribution we invoke the Boltzmann Machine learning rule [10, 14, 9]. The objective of the Boltzmann Machine learning rule is to minimize the Kullback information distance between a target distribution $P_{w^*}[S|d]$, (of which the training set is a finite sample), and the distribution $P_w[S|d]$ sampled by the current (stochastic) network with parameters w.

The learning rule is formulated for a general system specified in terms of inputs, hiddens, and outputs: (x, h, y). Since the states of the hidden units are unknown for the learning examples, we compare the marginal distributions, i.e. the distributions integrated over the hidden variables:

[1]A distribution of the form $P(x) = Z^{-1}exp(-E(x)/T)$, where $E(x)$ is a cost-function, bounded from below, and T is a parameter

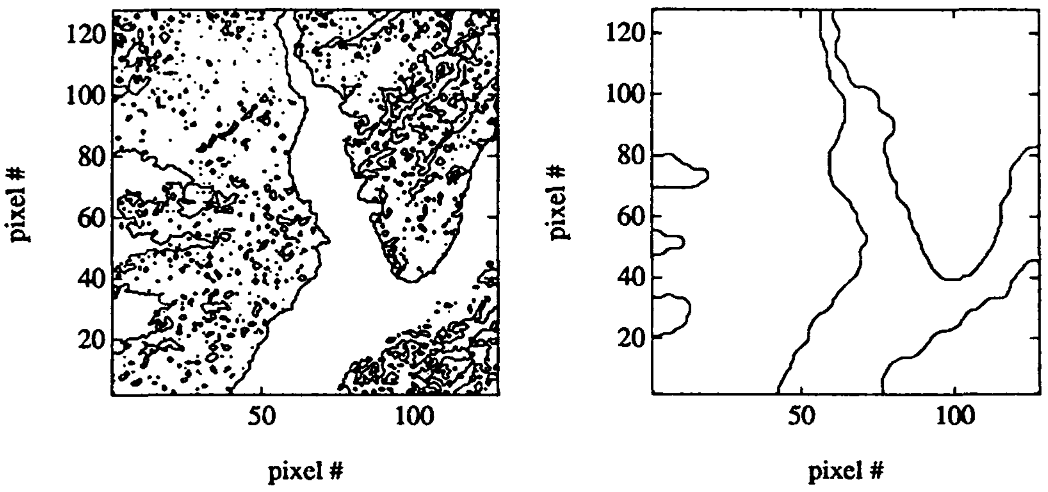


Figure 1: Subsampled, 128x128 pixel, preprocessed, Landsat image providing the raw (thresholded) evidence for water in the Igaliko region of Greenland (left), and the output evidence obtained by the unsupervised cellular neural network (right)

$$D[P_{w^*}, P_w] = \int Dx \int Dy \int DhP_{w^*}[x,h,y] \log \frac{\int DhP_{w^*}[x,h,y]}{\int DhP_w[x,h,y]} \tag{6}$$

$\int Dx$ is short for $\int \prod_j dx_j$. The learning algorithm is derived by gradient descent minimization of the information distance. The recursive learning algorithm reads:

$$w_\nu^{n+1} - w_\nu^n = \eta\left(\langle \Omega_\nu(x,h,y)\rangle\rangle_{clamped} - \langle \Omega_\nu(x,h,y)\rangle\rangle_{free}\right) \tag{7}$$

for a any cost-function linear in the parameters w_ν: $E(x,h,y) = \sum_\nu \Omega_\nu(x,h,y)w_\nu$, where $\Omega_\nu(x,h,y)$ is an expression in the stochastic variables. $\langle ...\rangle_{clamped}$ indicates that the average is performed with respect to the current Gibbs distribution, with *fixed* values for all the stochastic variables that are specified as ***input or output*** variables in an example of the database, i.e. (x,y). Similarly $\langle ...\rangle_{free}$ is the average with only ***input*** variables fixed. In brief we can characterize the learning process as follows: the parameters are adjusted to minimize the difference between the correlations in situations with and without the teacher specifying the correct output [10, 9].

We can employ the above formalism for unsupervised learning if we let our output units S play the role of hidden units and partition the input image d into a Boltzmann "input" part and a Boltzmann "output" part. The Boltzmann learning process then adapts the model until we reach a parameter set for which the "output" part of the image is estimated correctly from "input" part. Our procedure can be viewed as an example of statistical *cross-validation.*

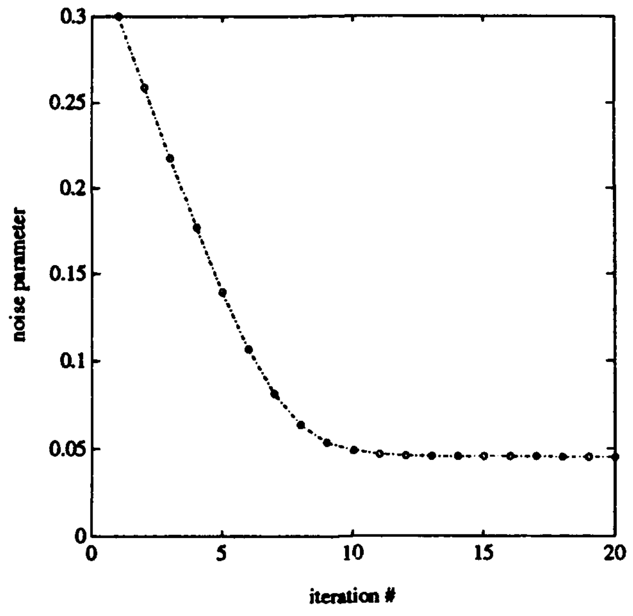


Figure 2: Unsupervised Boltzmann learning of the noise parameter of the cellular segmentation network.

4 Experimental and concluding remarks

Our case-study concerns the segmention of a subsampled Landsat satellite image. The input signal is a preprocessed 128 * 128 pixel image representing the evidence for water in the *Igaliko* region in Greenland. The preprocessing scheme establish the evidence using four frequency bands. Part of the image has been classified manually into five classes and the evidence is the result of a simple linear model relating the intensities in the four bands to the classification. Thresholding the evidence at zero results in Figure 1a). We adapt the noise parameter of the cellular network in unsupervised mode using a gradient descent parameter of 0.15. In Figure 1b) we show the segmentation of the adapted network. The convergence of the noise-parameter w_d towards the selfconsistent optimal value is presented in Figure 2.

In conclusion we have shown that the Boltzmann Machine learning rule can be used for identification of parameters in cellular neural networks designed using Bayesian reasoning. By invoking a crossvalidation-like procedure we were able to adapt the parameters of the cellular network without supervision, hence generalizing our earlier results on parameter estimation in cellular networks with hidden units [8].

Acknowledgement

This research is supported by the Danish Research Councils for the Natural and Technical Sciences through the Danish Computational Neural Network Center. I thank Bjarne K. Ersbøll for providing the Igaliko imagery.

References

[1] J. Besag: *Spatial Interaction and the Statistical Analysis of Lattice Systems.* J.Roy.Statist.Soc, **B36**, 192-236, (1974).

[2] J. Besag: *On the Statistical Analysis of Dirty Pictures.* J.Roy.Statist.Soc, **B48**, 259-302, (1986).

[3] P. Carnevali, L. Coletti and S. Paternello: *Image processing by simulated annealing.* IBM Journal of Research and Development **29**, 569-579, (1985).

[4] L.O. Chua and L. Yang: *Cellular Neural Networks: Theory.* IEEE Transactions on Circuits and Systems **35**, 1257-1272, (1988).

[5] J.B. Cole: *The Statistical Mechanics of Image Recovery and Pattern Recognition.* Am.J.Phys. **59**, 839-42, (1991).

[6] G.R. Cross and A.K. Jain: *Markov Random Field Texture Models.* IEEE Transactions on Pattern Analysis and Machine Intelligence **5**, 25-39, (1983).

[7] D. Geman and S. Geman: *Stochastic Relaxation, Gibbs distributions and the Bayesian restoration of images.* IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-6**, 721-741, (1984).

[8] L.K. Hansen: *Boltzmann Learning of Parameters in Bayes Visual Reconstruction.* Proceedings of the First Danish Conference on Pattern Recognition and Image Analysis. Ed.: S.I.Olsen. Department of Computer Science, University of Copenhagen, 92/8, (1992).

[9] J. Hertz, A. Krogh and R.G. Palmer: *Introduction to the Theory of Neural Computation.* Addison Wesley, New York (1991).

[10] G.E. Hinton and T.J. Sejnowski : *Learning and Relearning in Boltzmann Machines.* In Parallel Distributed Processing, vol. 1, chap 7. Eds. Rumelhart D.E. et al. Cambridge: MIT Press.

[11] B.R. Hunt: *Bayesian Methods in Nonlinear Digital Image Restoration.* IEEE Transactions on Computers **C-76**, 219-229, (1977).

[12] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi: *Optimization by Simulated Annealing.* Science **220**, 671-680, (1983).

[13] N. Metropolis et al.: *Equation of State Calculations by Fast Computing Machines.* J. CLem. Phys. **21**, 1087-1099 (1953).

[14] C. Peterson and J.R. Anderson: *A Mean Field Learning Algorithm for Neural Networks.* Complex Systems **1** 995-1019, (1987).

[15] M.W. Roth: *Survey of Neural Network Technology for Automatic Target Recognition.* IEEE Transactions on Neural Networks **1**, 28-43, (1990).

Learning State Space Trajectories in Cellular Neural Networks

Andreas J. Schuler, Peter Nachbar[*] and Josef A. Nossek
Institute for Network Theory and Circuit Design
Technical University of Munich, Germany
Email: ansc@nws.e-technik.tu-muenchen.de

Leon O. Chua
Electronics Research Laboratory, College of Engineering
University of California, Berkeley

Abstract

This paper presents a learning algorithm similar to the Backpropagation-Through-Time approach. The algorithm is based on the minimization of an error criterion, which is defined as the product of a function of the state at a given time and the integral of an entire time function of the state over the trajectory prior to this time. The technique of the calculus-of-variation will be used to evaluate the gradient of the error in the parameter space, which can be used to descend to a minimum on the error surface. We will adopt this theory to CNNs and show some simple examples of the learning of CNN parameters.

1 Introduction

We concentrate on a subclass of neural networks: *Cellular Neural Networks* (CNNs) [2, 7], where the interconnections between the neurons or cells are translationally invariant and only local. Therefore, the number of weights, which have to be learned or designed, is very small. Because of the piecewise linear transfer function of the CNNs the gradient algorithms of Almeida [1] and Pineda [9, 10, 11], which are backpropagation algorithms generalized for recurrent neural networks, can not be used. Based on the idea of Pearlmutter [8] to design the trajectory of recurrent dynamic neural networks, we will use the trajectory of the CNN state to gain information about the gradient of a modified error functional.

Other approaches towards the systematic design of CNNs have been proposed by Zou et. al. [16], and later by Slot and Kacprzak [15]. Chua and Thiran [3] provide a method for synthesizing CNNs for simple applications, and Seiler et. al. [14] show how to systematically design a CNN with stable and unstable outputs while simultaneously maximizing its robustness. Except for [3], which is restricted to relatively simple problems, the correct operation of the network is not guaranteed since only the stable or unstable patterns of the dynamical system are designed.

In the next section we will use the calculus-of-variations technique to minimize an error functional of a dynamical system in general. The derived equations will be applied to CNNs in Section 3, and some examples will be shown in Section 4.

2 The Backpropagation-through-Time Algorithm

The *Backpropagation-through-Time* (BTT) [8, 6] algorithm can be derived from solving a classical variation problem for any dynamical system [13], which is given by a system of differential equations

[*]Supported by a grant from the Ernst von Siemens foundation.

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, t, \mathbf{p}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the system state and $\mathbf{p}$ a parameter vector of the system.

The performance of the system is assumed to be measured by a scalar functional E which bookkeeps the errors over the whole transient, i.e. over the finite interval $[0, T]$:

$$E = \mathbf{L}_1^{\mathsf{T}}(\mathbf{x}(T)) \int_0^T \mathbf{L}_2(\mathbf{x}, t) dt. \tag{2}$$

$\mathbf{L}_1 : \mathbb{R}^n \to \mathbb{R}^n$ measures the error at the given time T, whereas $\mathbf{L}_2 : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is integrated over the given interval, to take into account the effect of a change of the parameters on the trajectory. This can be used to learn a specific trajectory as well.

The problem is to minimize the error functional (2) under the equality constraint given by (1). Using the vector equivalent of the Lagrange multiplier $\lambda \in \mathbb{R}^n$ the solution to the constrained problem can be obtained by minimizing

$$E' = \int_0^T \left[\mathbf{L}_1^{\mathsf{T}}(\mathbf{x}(T))\mathbf{L}_2(\mathbf{x}, t) + \lambda^{\mathsf{T}} \left(\mathbf{F}(\mathbf{x}, t, \mathbf{p}) - \dot{\mathbf{x}}\right)\right] dt. \tag{3}$$

A necessary condition for an extremum of E' is that the first variation of E' be zero:[1]

$$\delta E' = \int_0^T \left[\mathbf{L}_2^{\mathsf{T}} \frac{\partial \mathbf{L}_1}{\partial \mathbf{x}} \delta\mathbf{x}\Big|_T + \mathbf{L}_1^{\mathsf{T}}(\mathbf{x}(T)) \frac{\partial \mathbf{L}_2}{\partial \mathbf{x}} \delta\mathbf{x} + \lambda^{\mathsf{T}} \left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}} \delta\mathbf{x} - \delta\dot{\mathbf{x}}\right) + \left(\mathbf{F}(\mathbf{x}, t, \mathbf{p}) - \dot{\mathbf{x}}\right)^{\mathsf{T}} \delta\lambda\right] dt. \tag{4}$$

Since $\delta E'$ must equal zero independent of the first variation $\delta\mathbf{x}$, (4) yields the following equations:

The Euler-Lagrange equation:

$$\dot{\lambda} + \left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right)^{\mathsf{T}} \lambda + \left(\frac{\partial \mathbf{L}_2}{\partial \mathbf{x}}\right)^{\mathsf{T}} \mathbf{L}_1(\mathbf{x}(T)) = 0. \tag{5}$$

The associated transversality condition:

$$\lambda(T) = \left(\frac{\partial \mathbf{L}_1(\mathbf{x}(T))}{\partial \mathbf{x}}\right)^{\mathsf{T}} \int_0^T \mathbf{L}_2 dt \Bigg|_{t=T}. \tag{6}$$

The gradient $\frac{\partial E}{\partial \mathbf{p}}$ can be obtained by the differentiation of (2) with respect to $\mathbf{p}$:

$$\frac{\partial E}{\partial \mathbf{p}} = \int_0^T \mathbf{L}_2^{\mathsf{T}} dt \left(\frac{\partial \mathbf{L}_1}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right)\Big|_T + \mathbf{L}_1^{\mathsf{T}} \int_0^T \left(\frac{\partial \mathbf{L}_2}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right) dt. \tag{7}$$

With the partial derivative of (1) with respect to $\mathbf{p}$:

$$\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{p}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}} + \frac{\partial \mathbf{F}}{\partial \mathbf{p}}, \tag{8}$$

and using the definition of λ from (5) together with (6), (7) can be simplified to

$$\frac{\partial E}{\partial \mathbf{p}} = \int_0^T \lambda^{\mathsf{T}} \left(\frac{\partial \mathbf{F}}{\partial \mathbf{p}}\right) dt. \tag{9}$$

[1]The partial derivative of a scalar function $f(\mathbf{x})$ with respect to a vector $\mathbf{x}$ is defined to be a row vector

A necessary condition for an extremum of E is $\frac{\partial E}{\partial \mathbf{p}} = 0$.

Minima of the error functional can be found using various gradient techniques like conjugate gradient methods or Newton-like methods. The BTT-algorithm works as follows:

```
begin Minimizing the Error Functional $E$
  start with a random parameter vector $\mathbf{p}$
  repeat
    integrate $\dot{\mathbf{x}}$ from $t=0$ to $T$
    integrate $\dot{\lambda}$ from $t=T$ to $0$ and simultaneously evaluate the gradient
    perform a gradient step to reduce the value of the error
  until gradient of $E$ becomes zero
end.
```

3 The BTT-Algorithm for CNNs

The backpropagation-through-time algorithm can be applied to CNNs [2, 7], and the CNN equations (10) and (11) have to be used for (1):

$$\dot{x}_{c,d} = -x_{c,d} + \sum_{\nu=-r}^{+r} \sum_{\mu=-r}^{+r} \left[a(\nu,\mu) y_{c+\nu,d+\mu} + b(\nu,\mu) u_{c+\nu,d+\mu}\right] + i \,, \tag{10}$$

$$y_{c,d}(t) = f(x_{c,d}(t)) = \frac{1}{2}\left(|1 + x_{c,d}(t)| - |1 - x_{c,d}(t)|\right) \,. \tag{11}$$

The state variables $x_{c,d}$, where $1 \le c \le N$ and $1 \le d \le M$ define the two-dimensional grid, can be arranged in a state vector $\mathbf{x} \in \mathbb{R}^n$, with $n = NM$. The summations are restricted to the valid range of each index. The parameter vector $\mathbf{p}$ of a CNN is composed of the variables of the *cloning template* $a(\nu,\mu), b(\nu,\mu)$ and i, where $-r \le \nu \le +r$ and $-r \le \mu \le +r$ and r is the *neighborhood.*

The partial derivative of the state equations with respect to the state variables can be computed from the right hand side of (10):

$$\frac{\partial F_{c+\nu,d+\mu}}{\partial x_{c,d}} = \begin{cases} 0\,, & \text{if } |\nu| > r \text{ or } |\mu| > r\,; \\ -1 + a(0,0) f'(x_{c,d})\,, & \text{if } \nu = 0 \text{ and } \mu = 0\,; \\ a(-\nu,-\mu) f'(x_{c,d})\,, & \text{else}\,. \end{cases} \tag{12}$$

The partial differentiation of (10) with respect to the parameters yields:

$$\frac{\partial F_{c,d}}{\partial a(\nu,\mu)} = f(x_{c+\nu,d+\mu})\,, \qquad \frac{\partial F_{c,d}}{\partial b(\nu,\mu)} = u_{c+\nu,d+\mu}\,, \qquad \frac{\partial F_{c,d}}{\partial i} = 1\,. \tag{13}$$

With the use of λ, which has the same dimensions as $\mathbf{x}$, the gradient of the error becomes:

$$\frac{\partial E}{\partial a(\nu,\mu)} = \int_0^T \sum_{c=1}^{N} \sum_{d=1}^{M} f(x_{c+\nu,d+\mu}) \lambda_{c,d} dt\,, \tag{14}$$

$$\frac{\partial E}{\partial b(\nu,\mu)} = \int_0^T \sum_{c=1}^{N} \sum_{d=1}^{M} u_{c+\nu,d+\mu} \lambda_{c,d} dt\,, \tag{15}$$

$$\frac{\partial E}{\partial i} = \int_0^T \sum_{c=1}^{N} \sum_{d=1}^{M} \lambda_{c,d} dt\,. \tag{16}$$

If the vector valued functions $\mathbf{L_1}$ and $\mathbf{L_2}$ are choosen to be the squared errors of the corresponding cells

$$L_{1_{c,d}} = \frac{1}{2}\left(f(x_{c,d}(T)) - y^*_{c,d}\right)^2 \quad \text{and} \quad L_{2_{c,d}} = \frac{1}{2}\left(f(x_{c,d}(t)) - y^*_{c,d}\right)^2\,, \tag{17}$$

where $y^*_{c,d}$ is the desired output of the network for cell c, d, the error functional becomes

$$E = \sum_{c=1}^{N}\sum_{d=1}^{M} \frac{1}{2}\left(f(x_{c,d}(T)) - y^*_{c,d}\right)^2 \int_0^T \frac{1}{2}\left(f(x_{c,d}(t)) - y^*_{c,d}\right)^2 dt\,. \tag{18}$$

Therefore (5) and (6) become:

$$\dot{\lambda}_{c,d} = \lambda_{c,d} - \sum_{\nu=-r}^{+r}\sum_{\mu=-r}^{+r} a(-\nu,-\mu) f'(x_{c,d})\lambda_{c+\nu,d+\mu} - \left(f(x_{c,d}) - y^*_{c,d}\right) f'(x_{c,d})\frac{1}{2}\left(f(x_{c,d}(T)) - y^*_{c,d}\right)^2 \tag{19}$$

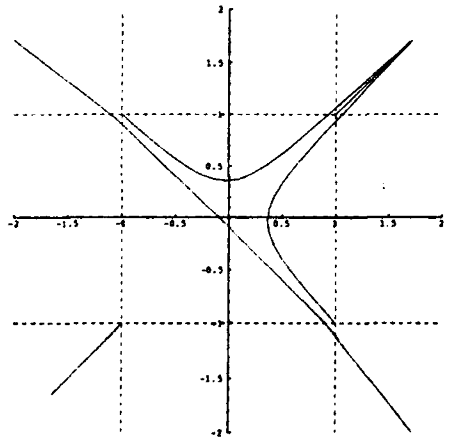
$$\lambda_{c,d}(T) = \left(f(x_{c,d}(T)) - y^*_{c,d}\right) f'(x_{c,d}(T)) \int_0^T \frac{1}{2}\left(f(x_{c,d}(t)) - y^*_{c,d}\right)^2 dt\,. \tag{20}$$

4 Examples

The examples in this section use linear extended CNNs, which are trained to do different tasks using the previous algorithm. In the case of several learning samples, the error and gradients are computed separately and then added for the whole sample. Throughout the following examples the initial parameter was set to zero: $\mathbf{p} = 0$, the time limit was $T = 10$ and the minimization of the error functional was done with the conjugate gradient minimization method of Fletcher and Reeves [4]. We will show the resulting template values, and as a measure of the computational effort of the minimization the number of function evaluations (running of the CNN forward and integrating the error over the transient), and the number of gradient evaluations (running backward in time while simultaneously computing the gradients).

4.1 State-based Logical OR

The design of a simple two-cell CNN which produces an output of both cells which is equal the logical "OR" of the initial state, where $+1$ is interpreted as 'true' and -1 as 'false' was reported in [14]. As there is no input of the CNN the operation is called state-based. The four different combinations of logical values were used as the learning samples. After three gradient steps with a total number of seven forward and backward integrations the error was zero. The resulting template values and the phase-plane trajectory of the four different initial conditions of this two-cell CNN are shown together with the boundary of the basins of attraction in Fig. 4.1. Although the self-feedback is slightly smaller than one, this template obtains a stable CNN. The symmetry of the templates is obtained automatically because of the symmetry of the problem.



$a =$	0.716	0.964	0.716
$b =$	0.000	0.000	0.000
$i =$	0.033		

Fig. 4.1: The phase-plane trajectories together with the boundary of the basins of attraction and the template values of the state-based logical OR CNN

4.2 Connected-Component-Detector

To learn the task of "Connected-Component-Detection" [5] we used all possible combinations of black and white cells in a linear CNN with five cells, which were the 32 learning samples. After only 13 gradient descents with 47 forward and backward integrations the error and the gradient were zero. In fig. (4.2) the learned template values for this problem are given.

$a =$	0.430	1.509	−0.430
$b =$	0.000	0.000	0.000
$i =$	−0.003		

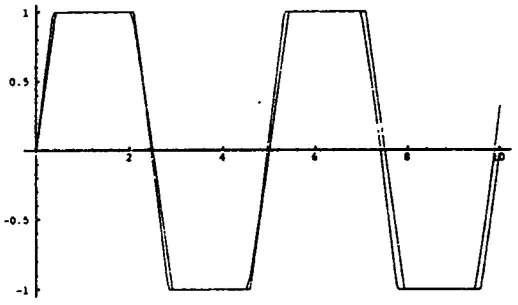
Fig. 4.2: The resulting template values of the Connected Component Detector

4.3 Two-Cell Oscillator

The last example will show that this algorithm can be used to train one cell of a simple two-cell CNN to follow a given trajectory. The desired trajectory of the output of cell 1 was a trapezoidal function of the time with an amplitude of 1, a period of 5 units of time and an absolute value of the slopes of 2. For this example, the functions $\mathbf{L}_1$ and $\mathbf{L}_2$ were chosen differently to measure only the error of cell 1 over the trajectory, so that the error functional E was

$$E = \frac{1}{2}\int_0^T (f(x_1(t)) - y_1^*(t))^2 dt\,. \tag{21}$$

Fig. (4.3) shows the trained output of cell 1, starting from the initial state $\mathbf{x} = (0, 0.7)^\mathrm{T}$ together with the corresponding values of the template.



$a =$	−0.672	1.197	4.000
$b =$	0.000	0.000	0.000
$i =$	0.006		

Fig. 4.3: The output of cell 1 of the two-cell oscillator and the corresponding template values

5 Conclusion

Our goal has been to find a learning algorithm for CNNs, which is able to learn the parameters of a CNN to perform a transition from an initial state to a state, which yields a prescribed output. We have shown a gradient technique which is able to learn the parameters such that a given error functional is minimized. Of course we find not always the global minimum, because the proposed algorithm uses the gradient of the error functional in the parameter space. Thus the algorithm stops, if the gradient is zero, whether this is a local minimum or a global one. Another drawback of the algorithm is the computational complexity. In order to compute the gradient of the error functional not only the differential equation of the CNN has to be integrated forward in time, but also the Euler-Lagrange equation, which is of the same dimensionality has to be integrated backward in time. Therefore the complete trajectory of the states has to be stored. This restricts the application of this algorithm to small examples. But in most cases this is not a severe restriction, as the learning can be achieved with relatively small samples.

References

[1] Luis B. Almeida. A learning rule for asynchonous perceptrons with feedback in a combinatorial environment. In *Proc. of the ICNN*, volume II, pages 609-618, San Diego, Calif., USA, 1987.

[2] L. O. Chua and Lin Yang. Cellular Neural Networks: Theory. *IEEE Tr. CAS*, 35:1257-1272, 1988.

[3] Leon O. Chua and Patrick Thiran. An analytic method for designing simple Cellular Neural Networks. *IEEE Trans. CAS*, 38(11):1332-1341, November 1991.

[4] R. Fletscher. *Practical Methods of Optimization*. Wiley + Sons, Chichester, second edition, 1989.

[5] T. Matsumoto, L. O. Chua, and H. Suzuki. CNN cloning template: Connected component detector. *IEEE Trans. CAS*, 37:633-635, 1990.

[6] Stefan Miesbach. Efficient gradient computation for continuous and discrete time-dependent neural networks. *Proc. of the ISCAS-91, Singapore*, pages 2337-2342, June 1991.

[7] J. A. Nossek, G. Seiler, T. Roska, and L. O. Chua. Cellular Neural Networks: Theory and circuit design. Technical Report TUM-LNS-TR-90-7, Technical University Munich, 12 December 1990.

[8] Barak A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263-269, 1989.

[9] Fernando J. Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59(19):2229-2232, November 1987.

[10] Fernando J. Pineda. Dynamics and architecture for neural computation. *Journal of Complexity*, 4:216-243, November 1988.

[11] Fernando J. Pineda. Recurrent backpropagation and the dynamical approach to adaptive neural computation. *Neural Computation*, 1:161-172, 1989.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Parallel Distributed Processing*, volume 1. M.I.T. Press, 1986.

[13] Andrew P. Sage. *Optimum Systems Control*. Prentice-Hall, Englewood Cliffs, N.J., 1968.

[14] Gerhard Seiler, Andreas J. Schuler, and Josef A. Nossek. Design of robust Cellular Neural Networks. Technical Report No. TUM-LNS-TR-91-13, Technical University Munich, September 1990.

[15] Krysztof Slot and Thomasz Kacprzak. Cellular neural network design problems for certain class of applications. In *Proc. of the ECCTD-91*, pages 516-523, Copenhagen, Denmark, 1991.

[16] Fan Zou, S. Schwarz, and J. A. Nossek. Cellular Neural Network design using a learning algorithm. In *Proc. of the first IEEE Int. Workshop on Cellular Neural Networks and their Applications, CNNA-90*, pages 73-81, December 1990.

SUPERVISED LEARNING OF THE STEADY-STATE OUTPUTS IN GENERALIZED CELLULAR NEURAL NETWORKS

Cüneyt Güzeliş
Faculty of Electrical-Electronics Engineering
Istanbul Technical University
80626 Maslak, Istanbul, Turkey
Tel. 90-1-276 36 16; Fax. 90-1-276 17 34
Email eecuneyt@tritu.bitnet

Abstract

It has been shown that the supervised learning of the steady-state outputs in a generalized Cellular Neural Network (CNN) is, in general, equivalent to a kind of constrained optimization problem. The objective function, also be called as the error function, is a measure of the distance between the sets of desired steady-state outputs and actual ones. The constraints are due to a set of design requirements which have to be met for providing the qualitative and quantitative properties for the network such as bipolarity of the steady-state outputs, complete stability etc. The approach presented in the paper uses the idea of the penalty function method in optimization theory where the constrained optimization problem is transformed into an unconstrained one by adding to the error function the terms corresponding to the constraints. A gradient descent algorithm has been proposed for solving the resulting unconstrained optimization problem. The learning algorithm developed generalizes the recurrent backpropagation algorithm in [6] into the generalized CNN which is a rather general class of dynamical neural networks including CNN, multi-layer Perceptron, continuous Hopfield network as special cases.

1 Completely Stable Generalized CNNs

The generalized CNN considered in this paper is a dynamical neural network recently introduced in [1]. It is indeed a generalization of CNN with respect to both connection topology and the circuit structure of the cells. As shown in Figure 1, each cell of a generalized CNN is an n th order dynamical circuit which is a cascade of a linear summation circuit, a linear dynamical circuit and an output nonlinearity. There are two kinds of connection weights in a generalized CNN : i) $w_{i,j}$ weighting the outputs y_j of the cells, ii) $z_{i,j}$ weighting the external inputs u_j. In this paper, the learning is assumed to be accomplished through modification of the connection weights only. For the sake of generality,

we define the input vector as $v = [v_u^T v_x^T]^T$. Where, $v_u = [u_1, u_2, ..., u_t]^T$, and respectively $v_x = [x_1^T(0), ..., x_t^T(0)]^T$ denotes the vector of external inputs, and respectively the vector of initial states with t is the number of cells. The network weight vector is defined as $W = [w_1^T, ..., w_t^T, z_1^T, ..., z_t^T]^T$. Where, $w_j = [w_{j,1}, w_{j,2}, ..., w_{j,Y_j}]^T$ is the weight vector of the j th cell such that its i th element belongs to the connection between the cell C_j and the i th neighbor of C_j ; and z_j s are defined similarly.
For a given input vector v , a generalized CNN with a chosen weight vector W will produce an output $y(t) \varepsilon R^t$. When the input vector is held fixed, the output $y(t)$ tends to a constant vector $y(\infty)$, called the steady-state output, if the network is completely stable [1]. Throughout the paper, we are interested in completely stable generalized CNNs only. Such generalized CNNs defines an algebraic map between the input and the steady-state output vector spaces. The complete stability of a network can be ensured by a set of conditions imposed on the connection weight vector W [1] , [2] , [5] , [8] . It is shown in [2] that a CNN is completely stable if the connection weights have the property of symmetry i.e. $w_{i,\hat{i}} = w_{\hat{i},i}$ for all i and $\hat{i}$. Such constraints constitute one part of the design constraints we mentioned and they can be described by the following set of nonlinear algebraic equalities and inequalities.

$$g_i(W) \leq 0 \qquad i \varepsilon \{1, 2, ..., C_1\} \tag{1}$$

In addition to the stability, some other qualitative or quantitative properties can be desired for a dynamical neural network. As proved in [2], the bipolarity of the steady-state outputs can be obtained by choosing self-feedback weights $w_{i,i}$ greater than a constant A_i . Such constraints constitute the second part of our design constraints and they can be described by the set of algebraic equalities and inequalities in (2).

$$h_i(W) \leq 0 \qquad i \varepsilon \{1, 2, ..., C_2\} \tag{2}$$

2 Supervised Learning as a Constrained Optimization Problem

The supervised learning of the steady-state outputs in a generalized CNN attempts to approximate an unknown input-(steady-state)output map $d = F(v)$ by minimizing an error function $E(W)$. The network is trained with the following set of pairs which are samples of the map $d = F(v)$:

$$\{(v^1, d^1), (v^2, d^2), ..., (v^N, d^N)\} \tag{3}$$

Where v^i and d^i represents the input and desired (steady-state) output for the i th sample, respectively. The error function $E(W)$ to be minimized is a measure of the difference between the desired and actual (steady-state) output sets. $E(W)$ is defined as the following summation of the instantaneous errors $E^i(W)$.

$$E(W) = \sum_{i=1}^{N} E^i(W) \tag{4}$$

Where $E^i(W) = D(y^i(\infty), d^i)$ denotes the distance between the vectors $y^i(\infty)$ and d^i in a vector space having a metric $D(y(\infty), d)$. Now, the supervised learning of the steady-state outputs in a completely stable generalized CNN can be formulated as a constrained optimization problem expressed in (5).

$$\min_{W\epsilon C} E(W) \tag{5}$$

Where, $C =: \{W \,|\, g_i(W) \leq 0 \;\; i\epsilon\{1,2,...,C_1\} \;\; and \;\; h_i(W) \leq 0 \;\; i\epsilon\{1,2,...,C_2\}\}$. The method preferred here for solving this minimization problem is the well-known penalty function method. The reason for using the penalty method is the possibility of converting the constrained optimization problem in (5) into an unconstrained one which can be solved by a gradient descent algorithm. The stochastic version of the learning algorithm developed in this paper reduces for unconstrained cases to the popular backpropagation algorithm which is indeed a stochastic gradient descent algorithm. The gradient descent method is proposed here for minimizing the augmented error function $\hat{E}(W)$ obtained by adding to the error function the penalty terms as in (6).

$$\hat{E}(W) = E(W) + \mu P(W) \tag{6}$$

Where $\mu > 0$ and $P(W)$ is a continuously differentiable function of W satisfying the conditions i) $P(W) \geq 0$ for all W , and ii) $P(W) = 0$ if and only if $W \epsilon C$ with C defined in (5) . It is known [13] that as μ goes to infinity any minimum point of the augmented error in (6) will converge to a solution of the constrained minimization problem described in (5).
The deterministic gradient descent algorithm proposed here is defined by the iterative algorithm

$$W(k+1) = W(k) - \epsilon(k)(\nabla_W \hat{E}[W(k)])^T \tag{7}$$

Where, $\nabla_W \hat{E}(W)$ is the gradient vector of $\hat{E}(W)$ with respect to W , and $\epsilon(k)$ is a nonnegative scalar minimizing $\hat{E}[W(k) - \epsilon(\nabla_W \hat{E}[W(k)])^T]$. It is known [13] that such an algorithm is globally convergent and its limit points are the minimum points of $\hat{E}(W)$. In the learning procedure proposed here, the coefficients are set to a small constant value ϵ , called the learning rate coefficient. One can select ϵ by trial and error. For large values of ϵ , the algorithm may not be convergent. Small values of ϵ may result in slow convergency. For the Euclidean metric and for a chosen penalty function, the gradient of $\hat{E}(W)$ can be given as

$$\begin{aligned}
\nabla_W \hat{E}[W(k)] &= \nabla_W E[W(k)] + \mu \nabla_W P[W(k)] \\
&= \nabla_W \sum_{i=1}^{N} \sum_{j=1}^{t} (y_j^i(\infty) - d_j^i)^2 + \mu \nabla_W \sum_{i=1}^{C_1} (max[0 g_i(W)])^2 \\
&+ \mu \nabla_W \sum_{i=1}^{C_2} (max[0, h_i(W)])^2 \\
&= \sum_{i=1}^{N} \sum_{j=1}^{t} 2\,(y_j^i(\infty) - d_j^i)\, \nabla_W y_j^i(\infty) + \mu \sum_{i=1}^{C_1} 2\, max[0, g_i(W)]\, \nabla_W g_i(W) \\
&+ \mu \sum_{i=1}^{C_2} 2\, max[0, h_i(W)]\, \nabla_W h_i(W)
\end{aligned} \tag{8}$$

3 The Stochastic Gradient Descent Algorithm

In the algorithm given above, the weights are adapted after the calculation of the gradient of the total error function by computing the contributions from all of the training samples.

Such a mode for learning is called batch learning. For large sets of training samples, the batch mode for learning needs much computation to be performed for every update of the weights. It might then be preferable to have an algorithm which updates the weights after every presentation of a traning sample. This well known idea of stochastic approximation leads us to develop the stochastic version of the algorithm in (7). In the stochastic algorithm, the training samples are viewed as random samples from an unknown distribution and the expected error function is estimated with the instantaneous error function $E^k(W(k))$ .Thus , the connection weight vector $W(k)$ is adapted according to

$$W(k+1) = W(k) - \epsilon(k)\left(\nabla_W \bar{E}^k[W(k)]\right)^T \tag{9}$$

Where $\bar{E}^k(W) = E^k(W) + \mu P(W)$ with $P(W)$ defined in (6). It is known [12] that stochastic approximation guarantees the convergence if the learning-rate coefficients $\epsilon(k)$ satisfies the conditions in (10) and (11).

$$\sum_{i=1}^{\infty} \epsilon(k) = \infty \tag{10}$$

$$\sum_{i=1}^{\infty} \epsilon^2(k) < \infty \tag{11}$$

The learning-rate coefficients $\epsilon(k)$ should decrease slowly in the sense of the divergent sum in (10) as well as quickly in the sense of the convergent sum in (11).

4 Conclusion

The supervised learning algorithm developed in this paper can be considered as a generalization of the recurrent backpropagation algorithm [6] to the generalized CNN where the connection weights can not be chosen freely due to some conditions imposed on them. The algorithm can be used not only for dynamical neural networks such as CNN, continuous Hopfield network but also for algebraic neural networks having some constraints on connection weights. The proposed algorithm is useful for training the CNN to perform image processing tasks.

References

[1] C. Güzeliş , and L. O. Chua , *"Stability analysis of generalized cellular neural networks," Accepted for publication in Int. J. Circuit Theory and Appl.*

[2] L. O. Chua, and L. Yang ,*" Cellular neural networks: Theory and Applications," IEEE Trans. Circuits Syst. Vol. 35 , No. 10, pp. 1257-1272, October 1988.*

[3] F. Zou, S. Schwartz, and J. Nossek ,*"Cellular neural network design using a learning algorithm," In Proc. IEEE Int. Workshop on Cellular neural networks and their applications, pp. 73-81, 1990.*

[4] L. O. Chua, and P. Thiran ,*" An analytic method for designing simple cellular neural networks," IEEE Circuits Syst. Vol. 38, No. 11, November 1991.*

[5] F. A. Savacı, and J. Vandewalle , " *On the stability analysis of cellular neural networks," IEEE Int. Workshop on Cellular Neural Networks and their Applications, 1992.*

[6] F. J. Pineda , " *Generalization of backpropagation to recurrent and higher order neural networks," in Neural Information Processing Systems, D. Z. Anderson Ed. New York : American Institute of Physics, pp. 602-611, 1988.*

[7] O. Farotimi, A. Dembo, and T. Kailath , " *A general weight matrix formulation using optimal control," IEEE Trans. Neural Networks, Vol. 2, No. 3, pp. 378-394, May 1991.*

[8] J. H. Li, and A. N. Michel, and W. Porod , " *Qualitative analysis and synthesis of a class of neural networks," IEEE Trans. Circuits Syst. Vol. 35, pp. 976-986, August 1988.*

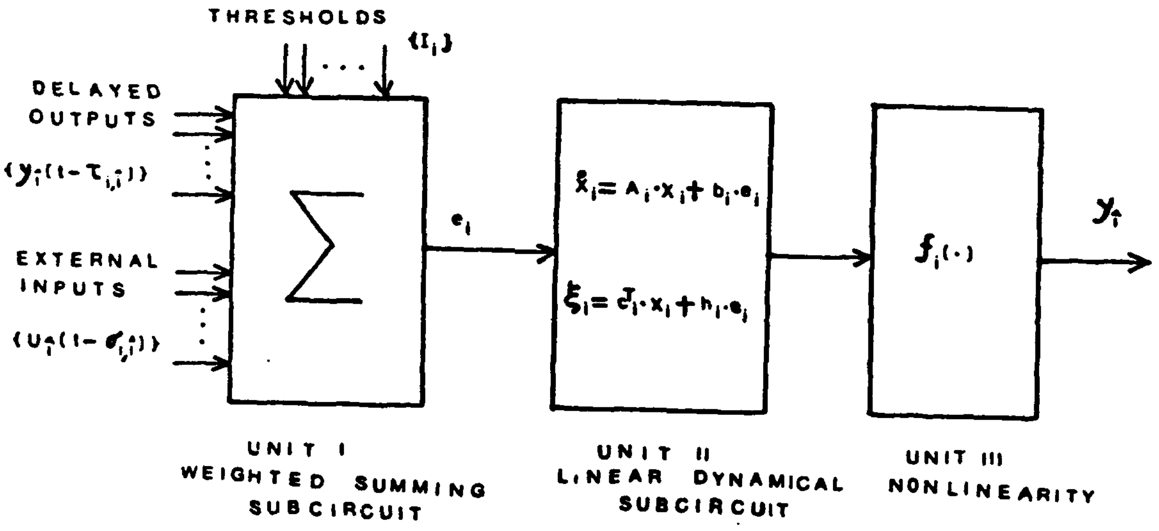
[9] F. M. A. Salam, and S. Bai , " *A new feedback neural network with supervised learning," IEEE Trans. Neural Networks, Vol. 2, No. 1, pp. 170-173, January 1991.*

[10] S. I. Sudharsanan, and M. K. Sundareshan , " *Equilibrium characterization of dynamical neural networks and a systematic synthesis procedure for associative memories," IEEE Trans. Neural Networks, Vol. 2, No. 5, pp. 509-521, September 1991.*

[11] T. Kohonen , *Self-organization and Associative Memory, 2 nd Ed., Springer-Verlag, New York, 1988.*

[12] B. Kosko , *Neural Networks and Fuzzy Systems, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.*

[13] D. G. Luenberger , *Introduction to Linear and Nonlinear Programming, Addison-Wesley, Massachusetts, 1973.*



THRESHOLDS
$\{I_i\}$
DELAYED OUTPUTS
$\{y_j(t-\tau_{i,j})\}$
EXTERNAL INPUTS
$\{u_j(t-\sigma_{i,j})\}$
$\sum$
e_i
$\dot{x}_i = A_i \cdot x_i + b_i \cdot e_i$
$\xi_i = c_i^T \cdot x_i + h_i \cdot e_i$
$f_i(\cdot)$
y_i
UNIT I
WEIGHTED SUMMING SUBCIRCUIT
UNIT II
LINEAR DYNAMICAL SUBCIRCUIT
UNIT III
NONLINEARITY

Figure 1: Block diagram of a cell.

Towards a Learning Algorithm for Discrete-Time Cellular Neural Networks

by Holger Magnussen and Josef A. Nossek

Institute for Network Theory and Circuit Design, Technical University of Munich

Abstract: *The learning process for a Discrete-Time Cellular Neural Network is formulated as an optimization problem. This involves minimizing an objective function, which is a measure of the errors in the desired input-to-output image mapping process performed by the network. With this approach, the learning algorithm finds the trajectories, so they no longer have to be designed by the user.*

1 Introduction

Discrete-Time Cellular Neural Networks (DTCNN) were introduced in [1]. The DTCNN is a clocked feedback network with translationally invariant weights. The transition from one clock cycle to the next is determined by the equations

$$x^c(k) =: \sum_{d\in N(c)} a^d y^d(k-1) + \sum_{d\in N(c)} b^d u^d + i$$

$$y^c(k) = g(x^c(k)) =: \begin{cases} +1 \text{ if } x^c(k) \geq 0 \\ -1 \text{ if } x^c(k) < 0 \end{cases}$$

where k is a positive integer corresponding to the clock period. $\mathrm{a} = (a^d)$ and $\mathrm{b} = (b^d)$ are the *templates* (weighted connections), i is the cell bias. $\mathrm{y}(k) = (y^c(k))$ is the output image of the net at time k, and $\mathrm{u} = (u^c)$ is the input image. $N(c)$ is a *neighborhood* of cell c. In this paper, the inputs are restricted to be binary-valued and constant, i.e. $u^d \in \{-1, 1\}$.

In most cases, neural networks are used to obtain some kind of mapping from input data onto output data. In contrast to this global point of view, the templates of CNNs and DTCNNs are usually derived from local rules at cell level: either only the fixpoints of the network or a detailed trajectory is designed by the user ([2], [3] or [4]). A Learning Algorithm for Supervised Training using a Noisy ERror surface for DTCNNs (LASTNERD) is introduced in this paper. It tackles the problem of learning the template coefficients from a different angle: an error measure for the mapping process of input images onto the desired output images is minimized. This (global) information is the only information which the algorithm uses, so that it can find a trajectory without the help of the user, and sets the templates accordingly.

The objective function is derived in the second section, and some prominent features of the resulting error surface are discussed in section 3. Section 4 describes the LASTNERD algorithm. Section 5 gives a short summary of the experiments which were performed to test the algorithm, and the last section summarizes the ideas of the paper and presents a short outlook.

0-7803-0875-1/92 $3.00 ©1992IEEE

2 The Objective Function

The process of learning is put into the form of an optimization problem, in which an objective function o, which depends on a parameter vector $\mathrm{v} = (\mathrm{a}, \mathrm{b}, i)$ containing the template coefficients, is minimized.

The goal of the learning process is to find the parameter vector $\mathrm{v}_{\mathrm{opt}}$ that guarantees an error-free mapping of a number of input images onto the corresponding desired output images. An objective function has to quantify the errors which the network makes in the mapping process. An obvious approach is the following: an image is fed into the network as the input image, the initial state is set to either the input image, to a white (-1) or to a black ($+1$) image. Then the network runs until it reaches a stable state. The Hamming distance between the converged output image and the desired output image corresponding to the input image is used as the error measure.

Oscillatory behavior can be a big problem for DTCNNs, so the case that the network does not reach a fixed point also has to be taken into account. The network is constantly monitored in order to detect cyclic behavior. Since a DTCNN is a clocked structure with binary outputs, this can be done in a fairly straightforward way. If oscillations are detected, the objective function punishes this behavior by taking on the maximum possible value. The optimization algorithm is then going to drive the parameter vector away from this (unstable) point in the parameter space towards regions with lower objective function values.

Let p denote a pair of an input image u and the corresponding desired output image $\mathrm{y}_d = (y_d^c)$. Then the modified Hamming distance $d_m(\mathrm{p}, \mathrm{v})$ for a given input-output image pair p at a certain point v in the parameter space becomes

$$d_m(\mathrm{p}, \mathrm{v}) = \begin{cases} \frac{1}{4M} \sum_{c=1}^{M} \left(y_\infty^c(\mathrm{u}, \mathrm{v}) - y_d^c\right)^2 & \text{if the net is stable} \\ 1 & \text{if the net oscillates} \end{cases} \tag{2}$$

where $0 \leq d_m \leq 1$. $y_\infty^c(\mathrm{u}, \mathrm{v})$ is the converged output of cell c, when the net was started with the input image u from the input-output image pair p, and y_d^c is the desired output of cell c in the output image of the input-output image pair p. M is the number of cells in the network.

In general, a neural network is required to learn not only one but a large number n of input-output image pairs p_i. Let $P = \{\mathrm{p}_1, \ldots, \mathrm{p}_n\}$ denote the set of all input-output image pairs which the network has to learn. An ideal objective function should average the modified Hamming distances $d_m(\mathrm{p}_i)$ over all n input-output image pairs p_i, but very often this is not advisable due to the computational cost.

To circumvent this problem, only a subset $S \subset P$ of all input-output image pairs is used in the averaging process. This subset S of size $l < n$ is selected randomly. Let $q = \{q_1, \ldots, q_n\}$ be a permutation of $\{1, \ldots, n\}$. Then the objective function $o(\mathrm{v})$ can finally be written as

$$o(\mathrm{v}) = \frac{1}{l} \sum_{i=1}^{l} d_m(\mathrm{p}_{q_i}, \mathrm{v}) \tag{3}$$

Again we have $0 \leq o(\mathrm{v}) \leq 1$. It is obvious that a multiple evaluation of the objective function with v fixed will result in slightly different values. This happens because different input-output image pairs p_i are used for the evaluation of $o(\mathrm{v})$ each time the function is called.

This random selection can also be viewed as a noise component in the objective function. This noise component has an important property: it vanishes, if a parameter vector $\mathrm{v}_{\mathrm{opt}}$ has been found, which guarantees a perfect input-to-output mapping of the network. In this case, the modified Hamming distance between the desired and the actual output will be zero, no matter which input-output image pairs $\mathrm{p}_i \in P$ are selected during the evaluation of the objective function.

The noise component in the objective function is crucial to the functioning of the algorithm. This will become obvious later in this paper. The amount of noise in the objective function can be influenced by the number of input-output image pairs used during the evaluation of the objective function. A ratio of $\frac{l}{n}$ close to one results in a small noise component, and a ratio close to zero in a strong noise component in the objective function.

3 The Error Surface

Before introducing the LASTNERD algorithm it makes sense to look at some peculiarities of the error surface, i.e. the objective function o as a function of the parameter vector v.

It is quite obvious that — due to the nonlinear (SIGNUM-) characteristic of each cell, which assures the binary nature of the cell output — the objective function is only going to change at those points in the parameter space where $x^c(k)$ in (1) is equal to zero.

Rewriting this condition into a more familiar form, we get

$$x^c(k) = \mathrm{v}^T \cdot \mathrm{e}^c(k-1) \overset{!}{=} 0 \tag{4}$$

where v is once again the parameter vector containing the all the coefficients of the a- and b-template and the bias i. The vector $\mathrm{e}^c(k-1)$ combines the respective cell output values $y^d(k-1)$ and values from the input layer u^d from the neighborhood of cell c at time $k-1$, and a constant $+1$ for the cell bias. Since the input values are restricted to be binary-valued in this paper, the vector $\mathrm{e}^c(k-1)$ will only have binary entries, thus

$$e^c_\mu(k-1) = \begin{cases} \pm 1 & \text{for } \mu = 1, \ldots, N-1 \\ 1 & \text{for } \mu = N \end{cases} \tag{5}$$

where N is the number of inputs of a cell and the dimension of v and $\mathrm{e}^c(k-1)$ (for an r-neighborhood on a square grid, $N = (2r+1)^2$ holds). (4) describes a hyperplane in $I\!R^N$, of which there can be as many as 2^{N-1} due to the constraints on the components of $\mathrm{e}^c(k-1)$ in (5).

Thus the error surface consists of a large number of convex polytopes R_i, which are bounded by the hyperplanes in (4). The objective function will be constant on these polytopes. Let ∂R_i denote the hull of polytope R_i. Hence, gradients on the error surface are either zero (for $\mathrm{v} \in R_i \setminus \partial R$) or undefined (for $\mathrm{v} \in \partial R$). In addition, the objective function is corrupted by the noise component that was explained in the preceeding section.

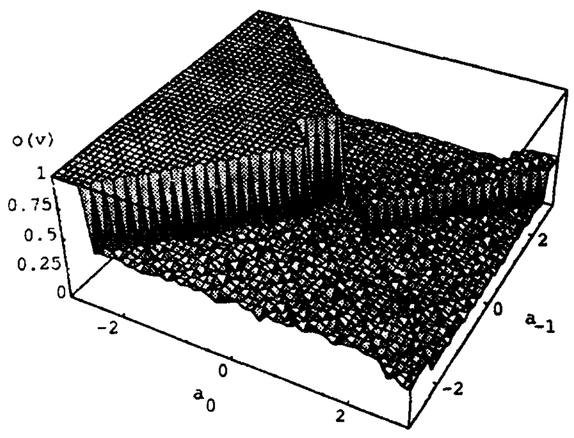


Fig. 1 A 2-dimensional cross-section through an error surface of a 3-dimensional example ($\mathbf{v}^T = (a_{-1}, a_0, a_1)$)

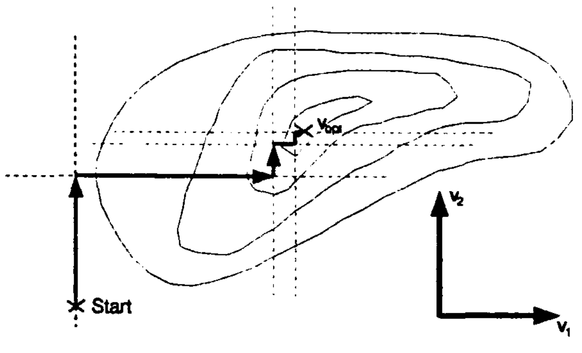


Fig. 2 A 2-dimensional example of the Alternating Variable method

Fig.1 shows a two-dimensional cross-section through a ($N = 3$)-dimensional error surface. The convex polytopes and the noise component of the error surface are clearly visible. Due to the binary nature of the vector $\mathbf{e}^c(k-1)$, the boundaries ∂R_i in a two-dimensional cross-section are always intersecting at right angles or run in parallel.

4 The LASTNERD Algorithm

For the reasons stated above, methods of optimization using gradient techniques like Quasi-Newton or Steepest Descent methods (see for example [5], [6]) are not applicable.

Direct Search methods are an alternative to gradient search strategies. They are usually more expensive in terms of the number of function evaluations, but on the other hand, they are much more flexible.

The LASTNERD algorithm is based on the well known Alternating Variable Method (as for example described in chapter 2 in [5]). In this method, line searches in parallel to the axes of the coordinate system are performed repeatedly. A line search procedure is supposed to find the minimum $o_{\min} = o(\alpha_{\min})$ of $o(\alpha) = o(\mathbf{v}_{\text{start}} + \alpha\mathbf{v})$, where α is a real number and $\mathbf{v}_{\text{start}}, \mathbf{v}$ are N-dimensional vectors (see section 2.6 in [6] for more details on line search algorithms). The result of each line search is then taken as a new (temporary) optimum. The whole process is repeated until the objective function falls below a desired level.

Fig.2 shows a simple two-dimensional example. In that figure, the thin solid lines are contour lines of a function defined over $I\!R^2$, and the thin dotted lines show the direction of the line searches.

Due to the special properties of the objective function, some modifications to the original Alternating Variable Method had to be made. The LASTNERD algorithm works as follows:

```
begin LASTNERD
   randomize the initial coefficients;
   repeat
     for for all coordinate axes do
     begin
       - Normalize the template coefficients;
       - perform a line search parallel to an axis of
         the coordinate system, starting from the current
         optimal point, and keep the result as the new
         optimum;
     end
   until template coefficients have been found
         which guarantee a zero (or sufficiently
         low) objective function;
end LASTNERD
```

To avoid any kind of cyclic behavior, the line searches along the different coordinate axes have to be performed in a random order.

The line searches remain the biggest problem in the algorithm. The function $o(\alpha)$ in the line search process is a noisy piecewise constant function, which will be very difficult to minimize. The way line searches are performed in the LASTNERD algorithm is by taking samples of $o(\alpha)$ from an interval I_α around $\alpha = 0$. Finding the best parameters for this sampling process is relatively difficult and requires some experience. If the distance between the samples is too small, than the algorithm becomes too expensive from a computational point of view. If on the other hand the samples are too far away from each other, the line search algorithm might jump over "trenches" in the objective function. The size of the interval around $\alpha = 0$ is another problem. The LASTNERD algorithm uses a compromise: If the objective function is still high, a larger interval I_α and a larger distance between the samples is chosen. As the objective function becomes smaller, the interval I_α becomes smaller and the sampling rate higher.

Another critical issue is the terminating condition for the algorithm. Due to the noise in the objective function, a zero objective function does not necessarily imply that the input-output mapping works for all image pairs. A better idea is to terminate the algorithm, when a number of consecutive evaluations of the error function have returned the value zero, thus making sure that a large number of input-output image pairs were tested during the evaluation of the objective function.

Any nonlinear optimization algorithm is usually prone to getting stuck in local minima. In simulations of the LASTNERD algorithm, it turned out that the noise component in the objective function is actually very helpful, since it may help the algorithm to jump out of local minima. Still, the amount of noise (controlled by the ratio $\frac{l}{n}$) has to be determined experimentally, because too much noise in the objective function might prevent the algorithm from descending to lower objective function values.

5 Experimental Results

The algorithm is working properly for moderate numbers N of template coefficients (around 20 to 30). While M, the number of cells of the neural network, influences only the execution time of the algorithm, simulations suggest that the performance of the LASTNERD algorithm deteriorates with increasing N, which corresponds to the dimensionality of the underlying optimization problem.

The algorithm was tested with well-known problems from literature, like Edge Detection, Connected Component Detection, Hole Filling, etc.. For these tasks, templates are known which solve the respective problem. Using a 10×5 network with a translationally invariant 1-neighborhood in a- and b-templates and a bias (19 template coefficients), the algorithm found templates solving the given problems.

The algorithm was finally applied to the problem of straight line (horizontal and vertical) detection, for which no known templates with a 1-neighborhood exist. The LASTNERD algorithm was not able to find templates for an error-free operation of the DTCNN, but it found a suboptimal solution resulting in a very low pixel error rate.

6 Conclusion and Outlook

This paper introduces the idea of mapping the learning task for DTCNNs onto a nonlinear optimization problem. The method used for solving this optimization problem — a modified Alternate Variable method — is relatively flexible, but not very efficient, since it does not take into account the specific structure of the DTCNN.

Still, the fact that the LASTNERD algorithm could find the templates for the examples shows that learning for DTCNNs can indeed be done in this way.

Currently, more advanced numerical algorithms, among them genetic algorithms, are applied to the inherent optimization problem. Using these algorithms, some interesting new applications for DTCNNs were found, which are to be reported in the near future.

Bibliography

[1] Hubert Harrer and Josef A. Nossek. Discrete-time cellular neural networks. Technical Report TUM-LNS-TR-91-7, Institute for Network Theory and Circuit Design, Technical University Munich, Germany, 1991.

[2] Gerhard Seiler, Andreas J. Schuler, and Josef A. Nossek. Design of robust cellular networks. Technical Report TUM-LNS-TR-91-13, Institute for Network Theory and Circuit Design, Technical University Munich, Germany, 1991.

[3] Fan Zou, Stephan Schwarz, and Josef A. Nossek. Cellular neural network design using a learning algorithm. In *Proc. International Workshop on Cellular Neural Networks and their Applications CNNA-90*, pages 73-81, Budapest, Hungary, December 1990.

[4] Hubert Harrer, Josef A. Nossek, and Fan Zou. A learning algorithm for discrete-time cellular neural networks. In *IJCNN'91 Proc.*, pages 717-722, Singapore, November 1991.

[5] W. Murray, editor. *Numerical Methods for Unconstrained Optimization.* Academic Press, 1972.

[6] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, 2 edition, 1987.

The Effective Weighting Method of CNN as a Recall

Hikaru MIZUTANI

Electrical Engineering, Shonan Institute of Technology

1-1-25 Tsujido-nisikaigan Fujisawa-city
Kanagawa, 258 Japan

1. Introduction

It is expected that neural network system such as Hopfield neural network (HF-NN) may solve the general associative and optimum problems. However, HF-NN's need m^2 synapses, where m is the number of neurons in the system. It is very difficult to fabricate such a large HF-NN. One of the method to solve the problem is to use Cellar Neural Networks (CNN) in which the number of synapses is determined by $dn/2$ where d is the number of synapses incident to each neurons. But there are synthesis problems to determine weights for association.

In this paper, we discuss a new synthesis method of CNN.

2. BN-mapping decision

We assume that the structure of CNN is fixed. In this section, we discuss the mapping from each bit of the data to a neuron. We call it a BN-mapping.

We assume that $\boldsymbol{h}_K$ is one of the n fundamental memories and that each fundamental memory $\boldsymbol{h}_K = [h_{1K} \cdots h_{mK}]$ consists of m elements, each of which is 1 or -1.

The energy function of full connection type NN is defined by the following equation[3]

$$E(\boldsymbol{x}) = -\boldsymbol{x}^t \mathbf{F} \boldsymbol{x}/2, \qquad (1)$$

where $\mathbf{F}$ is the connection matrix which is defined as

$$\mathbf{F} = \sum_{k=1}^{n} (\boldsymbol{h}_K \boldsymbol{h}_K{}^t - \mathbf{E}_m), \qquad (2)$$

where $\mathbf{E}_m$ is m dimensional unit matrix and the super subscript t represents the transport of a matrix. When the full-connection type NN recall j-th fundamental memory, a energy E_J is presented by

$$E_J = -\boldsymbol{h}_J{}^t \mathbf{F} \boldsymbol{h}_J/2. \qquad (3)$$

In the CNN which has local connection, the connection matrix $\mathbf{S}(\mathbf{F})$ can be represented by a sparse

0-7803-0875-1/92 3$.00 ©1992IEEE

matrix. Though, $\mathbf{F}$ is replaced $\mathbf{S}(\mathbf{F})$ which is sparse, we must select an optimal function $\mathbf{S}_i(.)$ from the set of $\{\mathbf{S}_i(.) \mid i=1, \cdots\}$. The decision depend on the BN-mapping. So, we must determine the BN-mapping so that the associated energy function of CNN can be decreasing as possible as we can. Because there are some fundamental memories, we must solve min-max problem such that the largest energy function can be minimized. That is the max-min problem represented by

$$u=\min(\max(-\boldsymbol{h}_j{}^t\mathbf{S}_i(\mathbf{F})\boldsymbol{h}_j/2 \mid j=1,n) \mid i=1,n).$$

Example

The CNN which is 7X5 array structure stores the 12 fundamental memories shown in fig. 2. The connection of each neuron is defined in fig. 1. By the proposed method, the BN-mapping is given as fig. 3.

3. Weight decision

3.1 Convergent radius

We modify the connection matrix $\mathbf{S}_i(\mathbf{F})$ to the following connection matrix for CNN;

$$\mathbf{N}=\sum_{k=1}^{n}\mathbf{A}_K(\mathbf{S}(\boldsymbol{h}_K\boldsymbol{h}_K{}^t-\mathbf{E}_m)), (4)$$

where

$$\mathbf{A}_K=\mathrm{diag}(a_{1K}, \cdots, a_{mK}). \quad (5)$$

Now, we assume that CNN can memorize a fundamental memory $\boldsymbol{h}_i$. When the output of CNN is $\boldsymbol{h}_i$, a input of the j-th neuron N_j is decided by

$$I_{ji}=\boldsymbol{t}_j\boldsymbol{h}_i+d_j. \quad (6)$$

where d_j is a threshold of a neuron N_j, then

$$h_{ji}=\mathrm{sign}(I_{ji}),$$

and $\boldsymbol{t}_j$ is j-th row element vector of matrix $\mathbf{N}$. If the prove vector or the output of HF-NN is $\boldsymbol{h}_i$ with r bit errors, the input of j-th neuron I_{ji}' is decided by

$$I_{ji}-2rs \leqq I_{ji}' \leqq I_{ji}+2rs, \quad (7)$$

where s is the biggest value in the absolute of the elements of matrix $\mathbf{N}$.

It is important that if the right side and the left side of equation (7) are equal in sign, CNN always correct r bit errors. It means that r is convergent radius of CNN or smaller than it.

In this paper, we decide such that $s=1$, and we must determine $\{\mathbf{A}_i, d_i \mid i=1,m\}$ so that r equals to r_{max} which is the maximum value of r.

Note that we can decide s. Because sign(.) cares only the sign of its parameter.

3.2 Computing method of the weight

By modifying equation (7) and using $s=1$, the following equations are obtained

$$I_{ji}-2r\geqq 0 \quad | \; h_{ji}>0,$$
$$I_{ji}+2r\leqq 0 \quad | \; h_{ji}<0. \quad (8)$$

And they are modified to

$$h_{ji}(I_{ji}-2r)\geqq 0. \quad (9)$$

Condition (9) is applied to all fundamental memories and all neurons. I_{ji} is described as following equation

$$I_{ji}=(\sum_{k=1}^{n} a_{jk}(h_{jk}\mathbf{h}_k{}^t-\mathbf{e}_j))\,\mathbf{h}_i + d_j. \quad (10)$$

So, Condition (9) can be modified to

$$h_{ji}(\sum_{k=1}^{n} a_{jk}(h_{jk}\mathbf{h}_k{}^t-\mathbf{e}_j))\,\mathbf{h}_i + d_j)-2r\geqq 0. \quad | \; i=1,n, \; j=1,m \quad (11)$$

where $\mathbf{e}_j$ is j-th row element of $\mathbf{E}_m$. We can get the following conditions using $s=1$ and equation (6)

$$\sum_{k=1}^{n} a_{jk}h_{jk}h_{ik}-1\geqq 0, \quad | \; i、j=1、m, \; i\neq j \quad (12)$$

$$1-\sum_{k=1}^{n} a_{jk}h_{jk}h_{ik}\geqq 0, \quad | \; i、j=1、m, \; i\neq j \quad (13)$$

We have to get r_{max} on condition (11), (12) and (13) for learning. Because these equations are described by linear equations, we can use the linear mathematical programming, *for instance* Symplex method, to solve it.

3.3 Example

By the proposed method, the weights are given as fig. 4. We have simulated the associative operation of CNN by using the resulted weights. Fig. 5 shows the some output pattern with weights of *the equation (2)* and proposed weights. The CNN by the proposed weighting method recalls all fundamental memories, but the CNN by the using the weight determined the equation (2) can not recall it.

References

[1] L. O. Chua and L. Yang, "Cellular neural networks: Theory", IEEE Trans. Circuits Syst., vol. 35, pp. 1257-1272, 1988.

[2] L. O. Chua and L. Yang, "Cellular neural networks: Applications", IEEE Trans. Circuits Syst., vol. 35, pp. 1273-1290, 1988.

[3] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", Proc. Nat. Acad. Sci. USA, 81, pp. 3088-3092, 1984.

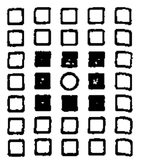


Fig. 1 The connection of CNN.

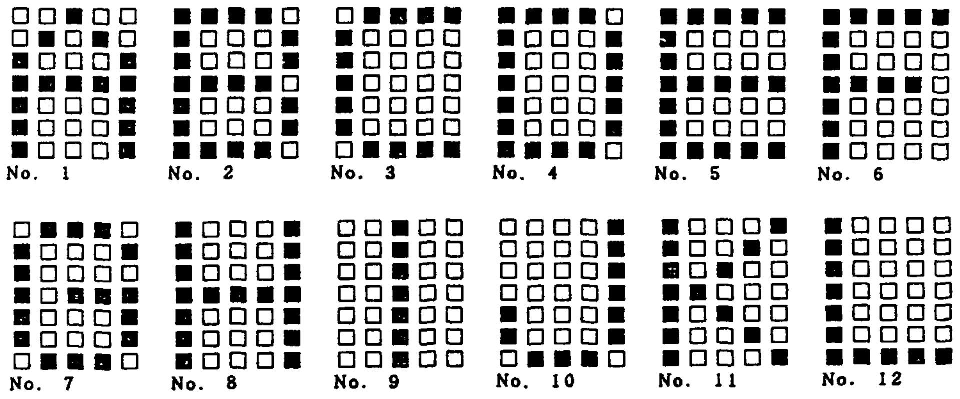


Fig. 2 Fundamental memories.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24
25	26	27	28	29
30	31	32	33	34

$\Downarrow$

0	9	32	4	26
5	31	22	7	33
11	14	25	13	23
15	1	27	30	24
17	18	20	12	29
21	19	16	8	28
6	3	2	34	10

Fig. 3 BN-mapping of CNN.

$$
W = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .5 & 0 & 0 & 0 & 0 & .5 & 0 & -.5 & 1 & -.5 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & C & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & C & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
-0.5 & 0 & 0 & 0 & -.5 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -0.5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -.25 & 0 & .5 & 0 & 0 & 0 & -.5 & .5 & 0 & 0 & -.25 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -.5 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & .5 & 0 & 0 & -.5 & 0 & .5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -.5 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & .5 & -1 & .5 & 0 & 0 & 0 & 0 & -.75 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .75 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\qquad
I = \begin{bmatrix}
-1.0 \\ 2.0 \\ -1.0 \\ 0.0 \\ 1.0 \\ 1.0 \\ -2.0 \\ -1.0 \\ 2.0 \\ 0.0 \\ 2.0 \\ -10000.0 \\ -1.0 \\ -10000.0 \\ 2.0 \\ 2.0 \\ 0.0 \\ -1.0 \\ 1.0 \\ 2.0 \\ 4.0 \\ -10000.0 \\ 1.0 \\ -10000.0 \\ 0.5 \\ 2.0 \\ -10000.0 \\ 0.0 \\ 3.0 \\ 0.5 \\ 1.0 \\ 0.0 \\ 1.0 \\ 0.0 \\ 2.0
\end{bmatrix}
$$

Fig. 4 The weights of CNN.

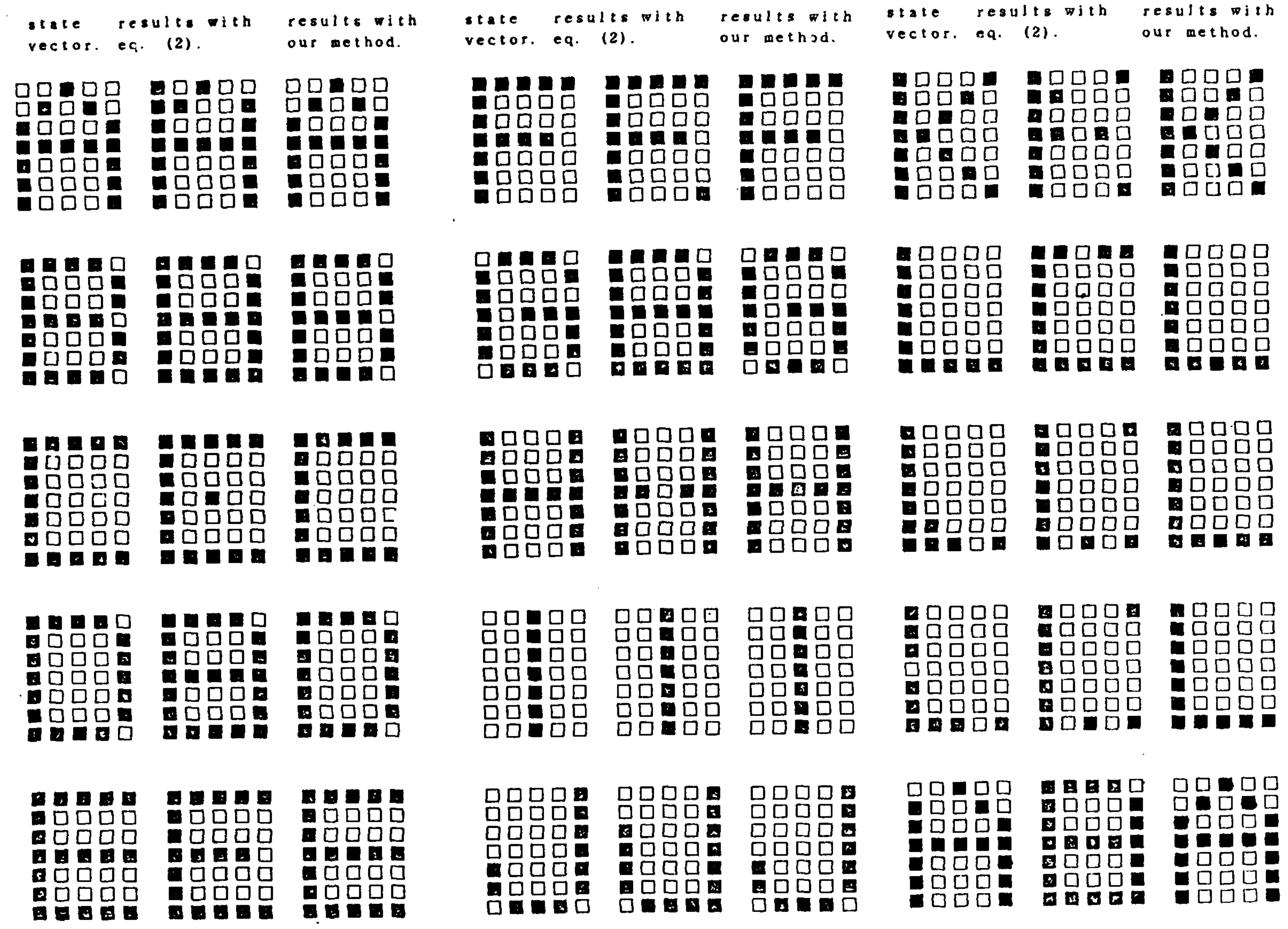


Fig. 5 The outputs of CNN.

CNN's AS INFORMATION PROCESSING DEVICES: TOWARDS A SYSTEMATIC APPROACH

Valerio Cimagalli
University of Rome "La Sapienza"
Dept. of Electronic Engineering
Via Eudossiana, 18 - I-00184 Roma, Italy
Tel. +39-6-44585836; Fax +39-6-4742647
E-mail: cima@tce.ing.uniroma1.it

Abstract - Among Artificial Neural Networks, CNN's have experienced an unprecedently known explosion of researches and applications, due to their unique property of using only local connections. So their use has become attractive for many purposes. We will just recall:

1. Modeling the physiology of the brain.
2. Modeling the processes of learning and classifying by human beings and animals.
3. Modeling the behavior of large physical systems, usually studied by means of statistical mechanics.
4. Using them as a new paradigm for very powerful electronic computers, well suitable to be implemented using VLSI.

As engineers, we are mainly interested in the last item, but it is important to not forget the strong links that exist between this item 4) and the preceding ones. Considering such links could make easier finding solutions to the many questions still open in the systematic design of CNN's.

The purpose of this talk would be starting a fruitful discussion on the answers to the following three questions:

1. What CNN's are? (i.e. definition)
2. How do CNN's operate? (i.e. behavior)
3. How they store information?

DEFINITION - No matter to say that everyone of us has a well defined idea of what a CNN is. But it is to notice that the many contributions that appeared in the literature after the first papers describing the pioneering work carried out at the University of California, Berkeley, generally do not comply with the definition given in those early papers. So we will try to investigate in a systematic way which items are essential properties of CNN's. As a conclusion it seemed us logic to identify the following three:

1. A regular array (not necessary bidimensional) of identical cells.
2. Links limited to a very restricted neighborhood.
3. Dynamics of each cell described by a nonlinear differential equation.

BEHAVIOR - Regardless of the particular kind of neural network under consideration, we may look at it as to a black box with an input and an output. Its purpose is to process the information contained in the input signal in order to obtain some specific result as, e.g., to recognize patterns, to classify data, to detect moving objects, to solve a problem of minimum, etc. This means that the network maps the space of all the admissible inputs into the space of its outputs according to some well defined rule.

In the said operation we may notice that many different circumstances can occur, such as, e.g.:

1. The network is a dynamic system described by a (vector) differential equation (autonomous or non autonomous).
2. The network implements an equation that can be explicitly solved with respect to the output, that turns out to be a function or a functional of the input.
3. Parameters of the equation may be constant or not.
4. Input signal may be fed through the initial state, or the forcing function, or the parameters.
5. Output may be drawn in many ways. If the output is an attractor, it may be of different kinds, even strange.

We will try to pick up connections between the different ways of operation and the definition of CNN's, as far as it is relevant to their behavior in storing and processing information. INFORMATION - Programmability is one of the most attractive properties of CNN's, but how to choose the most suitable network and how to program it in order to perform a given task, is still an open question. A step towards its solution could be to have an insight into the kind and the amount of information that a CNN is able to store. In fact it is well known that, due to the distributed way of storing and processing information, the organized structure of a neurocomputer adds significant relations to data fed to it. E.g., in the case of programming by examples, this is the reason why it is able to generalize from a limited number of training inputs.

Elsewhere we introduced the concept of "relational information" as a peculiar property of neurocomputers and suggested methods for measuring it. As a conclusion of this talk we will start discussing how such a concept can be applied to CNN

SOME STABILITY PROPERTIES OF CNN'S WITH DELAY

Pier Paolo Civalleri and Marco Gilli
Politecnico di Torino
Dipartimento di Elettronica
Cattedra di Elettrotecnica
C. Duca degli Abruzzi 24, I-10129 Torino, Italy;
Tel. 39-11-5644066/4096; Fax 39-11-5644015;
Email civalleri@itopoli.bitnet; gilli@itopoli.bitnet

ABSTRACT

In this paper some stability properties of cellular neural networks with delay (DCNN's) are studied; reciprocal and non-reciprocal DCNN's are considered. It is shown that a symmetric DCNN can become unstable if the delay is suitably chosen; moreover a sufficient condition is presented to assure the complete stability: such a condition establishes a relation between the delay time and the parameters of the network. A sufficient condition for the complete stability is also given for non-reciprocal networks: such a condition is independent from the delay time and depends only on the cloning-template and the delay-cloning-template.

1. INTRODUCTION

Cellular neural networks (CNN's), introduced by L. O. Chua and L.Yang in 1988 [1], [2], have found important applications in signal processing, especially in static image treatment. The stability of such networks has been investigated in [1], [3] and [4]: in [1] it has been proved that symmetric CNN's are completely stable; in [3] a weaker property (stability almost everywhere) has been established for the class of the positive cell linking templates and this result has been extended in [4], by means of equivalent transformations.

Processing of moving images requires the introduction of delay in the signals transmitted among the cells (DCNN's) [5]. The study of stability in this case is much more difficult than for conventional CNN's. In [6] it has been proved that positive cell linking templates are stable almost everywhere. In this paper we use the technique of the Lyapunov functionals to study the dynamic behaviour of two types of DCNN's. Firstly we consider a symmetric CNN and suppose to introduce a delay among the cells; we will prove that a symmetric CNN with delay can become unstable if the delay is suitably chosen and we will give a sufficient condition in order to ensure complete stability. This condition relates the parameters of the network to the introduced delay τ. Then we study a general non reciprocal DCNN, with the constraint that any signals interchanged between a couple of cells is delayed, while the feedback of a cell on itself is not delayed: in this case we derive a sufficient condition to ensure the complete stability of the network.

0-7803-0875-1/92 $3.00 ©1992IEEE

2. CELLULAR NEURAL NETWORKS AND CELLULAR NEURAL NETWORKS WITH DELAY

In order to study the stability properties of a DCNN, let us compare the state equation of a CNN (Cellular neural networks without delay) with the one of a DCNN (Cellular neural networks with delay). Given a $M \times N$ CNN, after having ordered the cells in some way, the state equation can be written as follows:

$$\dot{x} = -x + Ay + Bu \tag{1}$$

where:

x, $\dot{x} \in R^{M\times N}$ are the state vector and its derivative;
$y \in R^{M\times N}$ is the output vector;
$u \in R^{M\times N}$ is the input vector;
A, $B \in R^{M\times N, M\times N}$ are matrices, derived from the feedback template and from the input template [5].

Equations (1) is a system of differential equations of the type:

$$\dot{x} = f(x)$$

where f is a mapping from $R^{M\times N}$ to $R^{M\times N}$.

The state equations of a τ-delayed DCNN, by proper ordering of the cells, can be written as:

$$\dot{x}(t) = -x(t) + A_0 y(t) + A_1 y(t-\tau) + Bu \tag{2}$$

where A_0, A_1, $B \in R^{M\times N, M\times N}$ are matrices, derived from the feedback template, the delay cloning template and the input template [5].
Equation (2) is a particular case of the general functional differential equation [11]:

$$\dot{x} = f(x_t)$$

$f \in C(C([-\tau, 0], R^{M\times N}), R^{M\times N})$ is a continuous mapping of the space of functions x_t into $R^{M\times N}$.

We study the stability properties of DCNN's described by equation (2). To simplify the proofs, we assume that $u = 0$.

Definition 1: A dynamical autonomous system is said to be completely stable if and only if, for any initial point in the state space, the (unique) forwards trajectory reaches a stable equilibrium point.

According to our previous notations, the state space is $R^{M\times N}$ for a system without delay and $C([-\tau, 0], R^{M\times N})$ for a system with τ-delay.

3. SYMMETRIC DCNN'S

The complete stability of a symmetric CNN without delay, described by equation (1), has been proven in [1] by introducing a suitable Lyapunov function, i.e. a mapping of $R^{M\times N}$ into R. When equation (1) is replaced by equation (2), such a function can be replaced by a suitable Lyapunov functional, i.e. by a mapping of $C([-\tau,0],R^{M\times N})$ into R [11].

Definition 2: We define the following Lyapunov functional:

$$V(x_t) = y'(t)Py(t) - \int_{-\tau}^{0}[y'(t+\theta)-y'(t)]A_1'f(\theta)A_1[y(t+\theta)-y(t)]d\theta \qquad (3)$$

where $f(\theta)$ is any scalar function continuous with its derivative on $[-\tau,0]$, $f \in C^1([-\tau,0],R)$ and $P = -I + A_0 + A_1$, A_1' is the transpose of A_1.

By means of such a functional, we can find a sufficient condition for the complete stability of DCNN's in terms of the euclidean norm of A_1, which we denote as $\|A_1\|$, and of the delay τ.

Theorem 1: If A_1 is invertible, P is symmetric and:

$$\|A_1\| < \frac{2}{3\tau} \qquad (4)$$

then the corresponding DCNN is completely stable.

Proof: see [12].

Moreover it is possible to find a symmetric DCNN and a delay τ, where the above condition is not verified, which can oscillate.
Consider the following dynamic equations of a (2×1)-DCNN:

$$\frac{dx_1}{dt} = -x_1(t) + a_{11}^0 y_1(t) + a_{12}^0 y_2(t) + a_{11}^1 y_1(t-\tau) + a_{12}^1 y_2(t-\tau)$$

$$\frac{dx_2}{dt} = -x_2(t) + a_{12}^0 y_1(t) + a_{11}^0 y_2(t) + a_{12}^1 y_1(t-\tau) + a_{11}^1 y_2(t-\tau)$$

Suppose the cells are working in linear region, i.e.:

$$\dot{x}(t) = Hx(t) + A_1 x(t-\tau) \qquad (5)$$

where

$$H = \begin{pmatrix} h & a_{12}^0 \\ a_{12}^0 & h \end{pmatrix}$$

$$A_1 = \begin{pmatrix} a_{11}^1 & a_{12}^1 \\ a_{12}^1 & a_{11}^1 \end{pmatrix}$$

and $h = a_{11}^0 - 1 > 0$.

Thus the eigenvalue equation for an arbitrary delay τ is:

$$\det(H - \lambda I + A_1\exp(-\lambda\tau)) = 0 \qquad (6)$$

It is possible to show (see [12]) that by choosing:

$$h > 0$$
$$0 < a_{12}^0 < \frac{1}{\tau}$$
$$a_{12}^1 < 0$$

the eigenvalue equation has one imaginary solution, while all the other infinite solutions have negative real part: this means that the system admits of a stable closed orbit and thus can oscillate. In [12] it is shown that in this case the condition stated in *Theorem 1* is not verified.
If all the parameters of the template and of the delay-template are positive, the system admits of a closed orbit, but this latter is not stable (this is the case of the cooperative-irreducible DCNN, dealt with in [6], where the stability almost everywhere can be assured).

4. NON RECIPROCAL DCNN'S

Let us consider a DCNN, described by state equations (2), where: $H = A_0 - 1 = hI$ is diagonal, A_1 is a general matrix (even non symmetric) and $u = 0$: this means that all the signals interchanged between the cells are τ-delayed, while the feedback of a cell on itself is not delayed.

Theorem 2: If the diagonal element of H, h, is greater than the sum of the moduli of the elements of the delay-cloning-template, then the corresponding DCNN is completely stable.

Proof: Let us introduce the following Lyapunov functional:

$$V(x_t) = x'(t)x(t) - \int_{-\tau}^{0} x'(t+\theta) \;\; c \;\; x(t+\theta)d\theta \tag{7}$$

where c is a scalar constant.

Suppose the system starts from an initial condition, $x_0(\theta) \in C([-\tau, 0], R^{M\times N})$ $\theta \in [-\tau, 0]$, such that for any $\theta \in [-\tau, 0]$ all the components of x_0 are less than 1; i.e. all the cells are working in the linear region. Note that if a cell reaches a saturation region, under the hyphotesis of theorem 2, it cannot leave this latter; thus the above hyphotesis can be done for the remaining cells.

The derivative of $V(x_t)$ yields the following expression until all the cells are working in the linear region:

$$\dot{V}(x_t) = [x(t), x(t-\tau)] \begin{bmatrix} 2H - c & A_1 \\ A_1' & c \end{bmatrix} \begin{bmatrix} x(t) \\ x(t-\tau) \end{bmatrix} \tag{8}$$

The above expression satisfies the following inequality:

$$\dot{V}(x_t) \geq [|x(t)|, |x(t-\tau)|] \begin{bmatrix} 2h - c & -\|A_1\| \\ -\|A_1\| & c \end{bmatrix} \begin{bmatrix} |x(t)| \\ |x(t-\tau)| \end{bmatrix} \tag{9}$$

The positivity of $\dot{V}$ is assured, by Silvester test, if:

$$2h - c > 0 \tag{10}$$

$$c^2 - 2hc + \|A_1\|^2 \leq 0 \ \rightarrow h > \|A_1\|. \tag{11}$$

It is easy to show that an upper bound of the norm of A_1 is given by the sum of the absolute values of the elements of the delay-cloning-template, which in the following will be indicated as S. By choosing a suitable value of c, the condition $h > S$ assures that the derivative of the Lyapunov functional is positive everywhere and this means that at least one cell will reach the saturation region. Since $h > S$, if the state of a cell x_i reaches the value 1 or -1, it remains in the saturation region, regardless of the values of the state of the other cells; therefore such a cell becomes a constant input for the system. In fact by supposing that the cell i reaches a saturation region, the state equation of the system can be written as:

$$\dot{x}^i = hIx^i(t) + A_1^{ii}x^i(t-\tau) + u^i \tag{12}$$

where:

$$x^i(t) = \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \cdot \\ \cdot \\ \dot{x}_{i-1}(t) \\ \dot{x}_{i+1}(t) \\ \cdot \\ \cdot \\ \dot{x}_{M\times N}(t) \end{pmatrix} \tag{13}$$

A_1^{ii} is obtained from A_1 by deleting the i-th row and the i-th column; the constant term u_i is obtained by multiplying the i-th column of A_1 by the saturation value of the i-th cell, $y_i = \pm 1$. By putting

$$z^i(t) = x^i(t) + (hI + A_1^{ii})^{-1}u^i \tag{14}$$

the state equation of the system assumes the form:

$$\dot{z}^i(t) = hIz^i(t) + A_1^{ii}z^i(t-\tau) \tag{15}$$

By using the same Lyapunov functional shown above it is possible to show that another cell must reach the saturation value and this procedure can be applied until all the cells have reached the saturation region. From this latter consideration, by using the same arguments of [1], the complete stability of the system can be derived. Note that the etsbilished condition of stability $h > S$ is valid also when a constant input is introduced in the state equation (2). QED

REFERENCES.

[1] L.O. Chua and L. Yang, "Cellular neural networks: Theory", *IEEE Transactions on Circuits and Systems*, Vol. 35, pp. 1257-1272, 1988.

[2] L.O. Chua and L. Yang, "Cellular neural networks: Applications", *IEEE Transactions on Circuits and Systems*, Vol. 35, pp. 1273-1290, 1988.

[3] L.O. Chua and T. Roska, "Stability of a class of nonreciprocal cellular neural networks", *IEEE Transactions on Circuits and Systems*, Vol. 37, pp. 1520-1527, 1990.

[4] L.O. Chua and C.W. Wu, "The universe of stable CNN templates",*Memo* UCB/ERL, *University of California at Berkeley*, April 1991.

[5] L.O. Chua and T. Roska, "Cellular Neural Networks with nonlinear and delay-type template elements", *Proc. CNNA-90*, pp. 12-25,1990.

[6] T. Roska, C.W. Wu, M. Balsi and L.O. Chua, "Stability of delay-type cellular neural networks",*Memo* UCB/ERL, *University of California at Berkeley*, November 1991.

[8] M.W. Hirsch, "Convergent activation dynamics in continuos time networks", *Neural Networks*, Vol. 2, pp. 331-349, 1989.

[9] M.W. Hirsch, "System of differential equations that are competitive and cooperative II: convergence almost everywhere", *SIAM J. Math. Anal.*, Vol. 16, pp. 423-439, 1985.

[10] H.L. Smith, "Monotone semiflows generated by functional differential equations ", *Journal of Differential Equations*, Vol. 66, pp. 420-442, 1987.

[11] J.K. Hale, *Theory of functional differential equations,* New York Springer, 1977.

[12] P.P. Civalleri, M. Gilli and L. Pandolfi: "On stability of Cellular Neural Networks with delay" *Report Politecnico di Torino*, November 1991.

Two Causes of Instability of Cellular Neural Networks

Patrick Thiran
Chair of Circuits and Systems
Department of Electrical Engineering
Swiss Federal Institute of Technology
CH-1015 Lausanne Switzerland
e-mail: thiran@elde.epfl.ch

Abstract

Since a Cellular Neural Network (CNN) is a nonlinear analog circuit, its solutions may have a complex dynamical behavior, from complete stability to chaos. This paper will show that this behavior may depend strongly on the boundary conditions set at the borders of the finite-sized CNN, the network being stable with some boundary conditions and unstable with others. We will then distinguish two kinds of completely unstable CNNs: those that are unstable because of the boundary conditions and those that are unstable because of the template that defines them regardless of the boundary conditions.

1 CNNs with boundary conditions

The architecture of CNNs is described in [1]. We will consider CNNs defined by a space-invariant A-template, without a B-template and with I = 0. They are therefore described by the state equation

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{m,n=-r}^{r} A_{m,n}\, y_{i+m,j+n}(t)$$

and the output equation $y_{ij}(t) = f(x_{ij}(t))$, where the piecewise-linear function $f(\cdot)$ is given by $f(v) = \frac{1}{2}(|v+1| - |v-1|)$ and r is the size of the neighborhood.

We will surround the rectangular array with *boundary cells*, whose outputs lie between -1 and 1 and remain constant with respect to time t. The width of this frame of boundary cells is equal to r. Their outputs will be referred to as the *boundary conditions* (Up to now, in most applications using CNNs, these boundary conditions have been set to zero most of the time).

2 Two kinds of completely unstable CNNs

A CNN is completely stable if every initial state converges to an equilibrium [2]. Likewise, a CNN will be said to be *completely unstable* if no initial state (except possibly a set of measure zero) does converge to an equilibrium.

Consider first a CNN with a neighborhood of size 1 (and cells arranged on a squared or hexagonal grid). Then we have the following result:

Theorem 1 *A CNN whose neighborhood size is 1 always has a stable equilibrium provided the boundary conditions are correctly chosen.*

For such a CNN, one can indeed show that at least one of the four configurations shown in figure 1 is always a stable equilibrium of the network. A black pixel corresponds to an output of 1 while a white pixel corresponds to an ouput equal to -1. This theorem is a corollary of a theorem proven in [3], which states that a CNN, with an array of cells of infinite size and an A-template such that $\sum_{m \text{ and } n \text{ even}} A_{m,n} > 1$, has always a stable equilibrium.

Consequently, if a CNN defined by a 3×3 template is completely unstable, we can modify these boundary conditions (by picking one of the four sets of boundary conditions represented in figure 1), so that this CNN does not remain unstable any longer. This implies that the complete instability of such a CNN is always due to the boundary conditions. It is possible however to find CNNs with a neighborhood of size 2 that are completely unstable, independently of the boundary conditions:

Theorem 2 *A one-dimensional CNN, of at least 37 cells, and defined by a template*

$$A = \begin{bmatrix} A_{-2} & A_{-1} & A_0 & A_1 & A_2 \end{bmatrix}$$

where $A_0 > 1$ *and*

$$\begin{aligned} A_{-2} + A_{-1} + A_0 + A_1 + A_2 &< 1 \\ A_{-2} - A_{-1} + A_0 - A_1 + A_2 &< 1 \\ A_{-2} - A_{-1} + A_0 + A_1 - A_2 &< 1 \\ -A_{-2} - A_{-1} + A_0 + A_1 - A_2 &< 1 \\ -A_{-2} - A_{-1} + A_0 + A_1 + A_2 &< 1 \end{aligned}$$

is completely unstable, regardless of the boundary conditions.

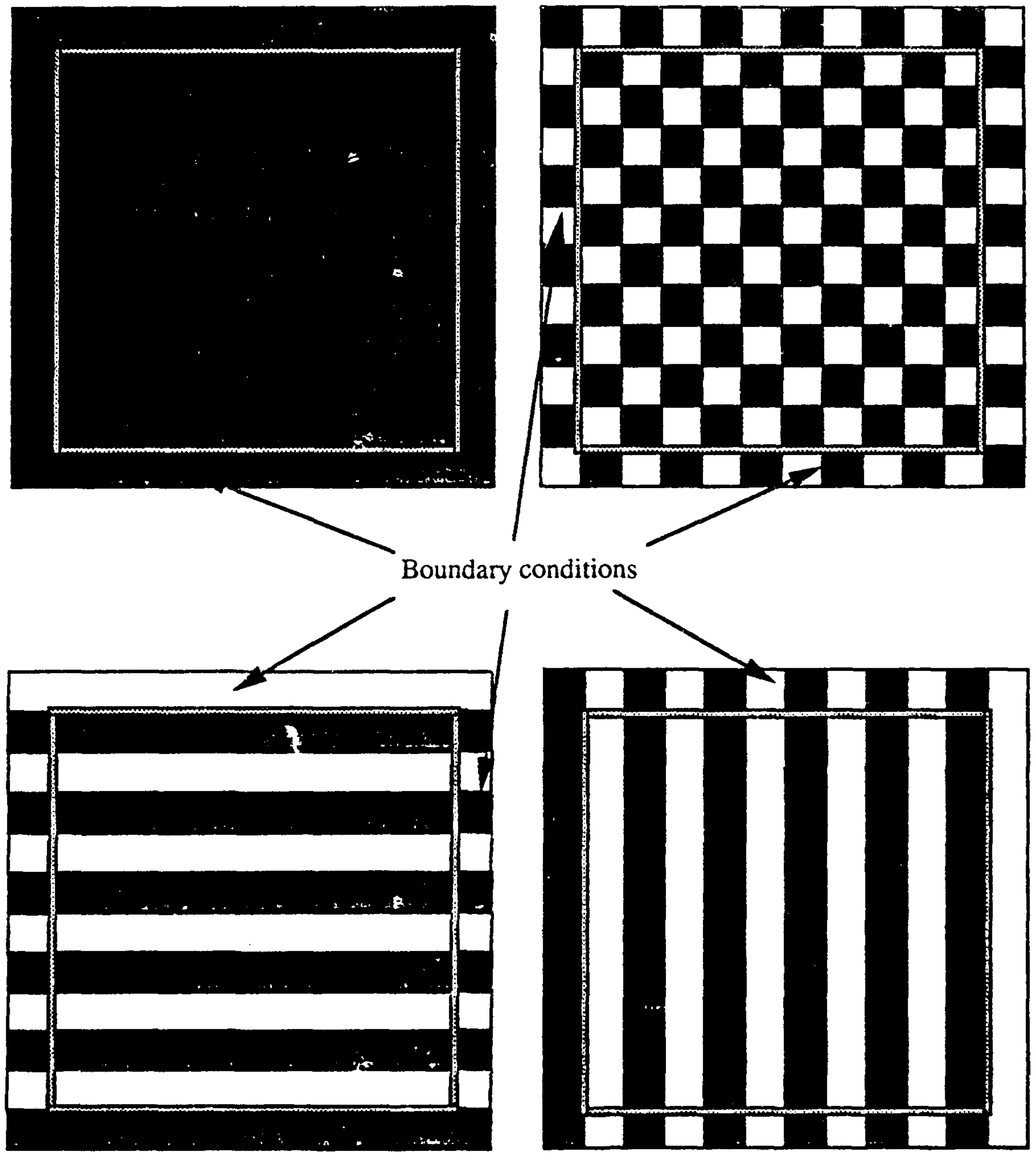


Figure 1: One of these four configurations is always an equilibrium of a CNN with a neighborhood size equal to 1.

The instability of this CNN is thus no longer due to the boundary conditions but is a characteristic of the template itself.

This theorem is also a corollary of a more general theorem proven in [3].

3 Examples

Example 1. The one-dimensional CNN defined by the template

$$A = \begin{bmatrix} -s & p & s \end{bmatrix}$$

is known to be completely unstable if $0 < p - 1 < s$ and if the two boundary conditions at both ends of the linear array are set to zero [4]. Nevertheless, since A is a 3×1 template, there exists a stable equilibrium if the boundary conditions are properly chosen, because of theorem 1. In fact, this happens when the boundary conditions are set to 1 or -1, as pointed out in [4] and [5]. Moreover, this CNN can be proven to be completely stable with such boundary conditions, if its number M of cells is less or equal than three. For $M = 1$ and $M = 2$, this can be easily proven. Take $M = 3$ and, for instance, the boundary conditions $y_0 = 1$ and $y_4 = -1$. The state equations describing this CNN are thus

$$\begin{aligned} \dot{x}_1(t) &= -x_1(t) + pf(x_1(t)) + sf(x_2(t)) - s & (1)\\ \dot{x}_2(t) &= -x_2(t) - sf(x_1(t)) + pf(x_2(t)) + sf(x_3(t)) & (2)\\ \dot{x}_3(t) &= -x_3(t) - sf(x_2(t)) + pf(x_3(t)) - s. & (3) \end{aligned}$$

Since $sf(x_2(t)) - s \leq 0$ and $-sf(x_2(t)) - s \leq 0$ for any time t, corollary 2 in [6] implies that if $x_1(t_0) < 0$ at some time t_0, then there is some time $t_0 \leq t_0' < +\infty$ such that $y_1(t) = -1$ for all $t \geq t_0'$, whereas if $x_3(t_1) < 0$ at some time t_1, then there is some time $t_1 \leq t_1' < +\infty$ such that $y_3(t) = -1$ for all $t \geq t_1'$. In these two cases, the network is equivalent to a CNN made of two cells, with boundary conditions equal to ± 1, which is completely stable.

We will now see that one of the two states $x_1(t)$ or $x_3(t)$ must become strictly negative at some time t_2. Suppose on the contrary that $x_1(t) \geq 0$ and $x_3(t) \geq 0$ for all time t. Then by adding (1) and (3), we get

$$\dot{x}_1(t) + \dot{x}_3(t) = -(x_1(t) + x_3(t)) + p(f(x_1(t)) + f(x_3(t))) - 2s \leq 2(p - 1 - s) < 0.$$

Hence $x_1(t_2) + x_3(t_2)$ becomes strictly negative at some time $0 \leq t_2 < \infty$, implying that $x_1(t_2) < 0$ or $x_3(t_2) < 0$. Consequently this CNN is completely stable.

Example 2. The one-dimensional CNN defined by the template

$$A = \begin{bmatrix} -s & v & p & s & 0 \end{bmatrix}$$

is also completely unstable if $p > 1$, $v > 0$ and $s > p - 1 + v$ and if the four boundary conditions are equal to zero, but has a stable equilibrium (every steady-state output being equal to 1) if the four boundary conditions are set to 1.

Example 3. Another way of introducing "boundary conditions" is to connect the first cell of a one-dimensional array to the last one, making a ring. If such a CNN is defined by the template

$$A = \begin{bmatrix} -s & p & 0 \end{bmatrix},$$

with $s > p - 1 > 0$, it is completely stable if its number of cells is even but completely unstable if its number of cells is odd! Note that without the additional connection between the first and the last cell, this CNN is completely stable for any number of cells, and for any values of the steady outputs of the boundary cells, since this template is acyclic [7].

Example 4. In contrast to the three previous examples showing CNNs whose stability depends on the boundary conditions, the CNN defined by the template

$$A = \begin{bmatrix} -p & p & p & -p & 0 \end{bmatrix},$$

where $p > 1$ is always completely unstable if its number of cells is sufficiently large, regardless of the boundary conditions, since it satisfies the set of inequalities of theorem 2.

4 Conclusion

The two theorems presented in this paper show that some CNNs are completely unstable because of the boundary conditions, and will not stay completely unstable if we change these boundary conditions properly; whereas other CNNs (with a neighborhood size greater or equal to 2) are always unstable regardless of the boundary conditions.

Acknowlegments

This work was supported by the Swiss National Science Foundation under Grant number 21-30287.90. The author would like to thank Prof. Martin Hasler for helpful discussions.

References

[1] L.O. Chua and L. Yang, "Cellular Neural Networks: Theory," *IEEE Trans. Circuits and Systems*, vol. CAS-35, pp. 1257–1272, 1988.

[2] L.O. Chua and T. Roska, "Stability of a class of Nonreciprocal Cellular Neural Networks", *IEEE Transactions on Circuits and Systems*, vol. CAS-37, pp. 1520–1527, 1990.

[3] P. Thiran, "Influence of Boundary Conditions on the Behavior of Cellular Neural Networks", submitted for publication, 1992.

[4] F. Zou and J. Nossek, "Stability of Cellular Neural Networks with Opposite-Sign Templates", *IEEE Transactions on Circuits and Systems*, vol. CAS-38, pp. 675–677, 1991.

[5] L. Vandenberghe and J. Vandewalle, "The Computation of Equilibrium Points in Cellular Neural Networks Using Complementary Pivoting", *Proceedings ECCTD-91*, Copenhagen, pp. 30–38, 1991.

[6] L.O. Chua and P. Thiran, "An Analytic Method for Designing Simple Cellular Neural Networks", *IEEE Transactions on Circuits and Systems*, vol. CAS-38, pp. 1322–1341, 1991.

[7] L.O. Chua and C.W. Wu, "On the Universe of Stable Cellular Neural Networks", Memo UCB/ERL M91/31, University of California at Berkeley, 1991.

Convergence of reciprocal Time-Discrete Cellular Neural Networks with continuous nonlinearities

N. Fruehauf [1], L.O. Chua [2], E. Lueder [1]

[1] Institut für Netzwerk u. Systemtheorie, Universität Stuttgart
Pfaffenwaldring 47, D-7000 Stuttgart 80 , Germany
Tel.: +49-711-685-7351; Fax: +49-711-685-7311
Email: norbert.fruehauf@ins.e-technik.uni-stuttgart.dbp.de

[2] Department of Electrical Engineering and Computer Sciences, UC Berkeley

Abstract
This paper outlines a proof for the convergence of reciprocal Time-Discrete Cellular Neural Networks with continuous, monotone increasing nonlinearities. The proof uses a Lyapunov function of the Time-Discrete Cellular Neural Network.

1 Introduction

The operation of every Cellular Neural Network (CNN) is based on dynamic transitions of the state variables which transform a given initial state into a desired output state. Consequently the analysis of CNN stability and convergence is a very important topic in CNN theory.

Up to now convergence for Time-Discrete CNNs with a threshold-type nonlinearity and eigendominant or single-neighbor dominant templates has been proved by Harrer and Nossek [1],[2]. Hui and Zak [3] proved the convergence of Time-Discrete CNNs with a piecewise linear output function and feedback operators which comply with:

$$A(i,j;i,j) - 1 > \sum_{\substack{k=0\\k\neq i}}^{M-1} \sum_{\substack{l=0\\l\neq j}}^{N-1} |A(i,j;k,l)| + \left| \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} B(i,j;k,l)u_{k,l} + d_{i,j} \right| \tag{1}$$

for all possible inputs $u_{i,j}$.

This paper outlines a proof for the convergence of reciprocal Time Discrete CNNs with continuous, monotone increasing nonlinearities. The required constraints for the feedback operator values are less restrictive than those given in equation 1.

2 Architecture of the Time-Discrete CNN

A two-dimensional and reciprocal Time Discrete CNN of $M \times N$ cells can be described by the following equations:

<u>**State Equation:**</u>

$$x_{i,j}(n+1) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} A(i,j;k,l) y_{k,l}(n) + \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} B(i,j;k,l) u_{k,l} + d_{i,j} \tag{2}$$

$$\text{for all} \quad i = 0,1,\cdots,M-1; \; j = 0,1,\cdots,N-1$$

Output equation

$$y_{i,j}(n) \quad = \quad f(x_{i,j}(n)) = \begin{cases} 1 & \text{for} \quad x_{i,j}(n) \geq 1 \\ \tilde{f}(x_{i,j}(n)) & \text{for} \quad |x_{i,j}(n)| \leq 1 \\ -1 & \text{for} \quad x_{i,j}(n) \leq -1 \end{cases} \tag{3}$$

$$\text{with} \qquad 0 < \frac{d\tilde{f}(x)}{dx} < \infty \quad \text{for} \quad |x| < 1 \quad \text{and} \quad \frac{d\tilde{f}(x)}{dx} = 0 \quad \text{for} \quad |x| = 1$$

Parameter assumptions

- Reciprocity of feedback operators

$$A(i,j;k,l) = A(k,l;i,j) \quad \text{for all} \quad i = 0,1,\cdots,M-1;\ j = 0,1,\cdots,N-1;\ k = 0,1,\cdots,M-1;\ l = 0,1,\cdots,N-1 \tag{4}$$

- Constraints for feedback operator values

$$A(i,j;i,j) \geq \sum_{\substack{k=0 \\ k\neq i}}^{M-1} \sum_{\substack{l=0 \\ l\neq j}}^{N-1} |A(i,j;k,l)| \quad \text{for all} \quad i = 0,1,\cdots,M-1;\ j = 0,1,\cdots,N-1 \tag{5}$$

Remark:

1. This constraint for the feedback operator values is sufficient but not necessary to guarantee convergence to the equilibrium states. It will be derived from a less restrictive constraint in paragraph 5.

3 Matrix-Vector Formulation

It is convenient to describe the dynamics of the Time-Discrete CNN with a matrix-vector notation which can be derived from equations (2) and (3) by ordering the output variables in a rowwise sequence:

$$X_\nu(n+1) = \sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu)U_\mu + D_\nu \tag{6}$$

and

$$Y_\mu(n) = f\left(X_\mu(n)\right) \tag{7}$$

with

$$\begin{aligned} X_{iN+j} &= x_{i,j} & (8)\\ Y_{iN+j} &= y_{i,j} & (9)\\ \mathcal{A}(iN+j, kN+l) &= A(i,j;k,l) & (10)\\ \mathcal{B}(iN+j, kN+l) &= B(i,j;k,l) & (11)\\ D_{iN+j} &= d_{i,j} & (12) \end{aligned}$$

$$\text{for all} \quad i = 0,1,\cdots,M-1;\ j = 0,1,\cdots,N-1;\ k = 0,1,\cdots,M-1;\ l = 0,1,\cdots,N-1 \tag{13}$$

Remark:

1. Observe that $\mathcal{A}(\nu,\mu)$ is symmetric (i.e. $\mathcal{A}(\nu,\mu) = \mathcal{A}(\mu,\nu)$) because of equations (4) and (10)

4 Convergence of the Time-Discrete CNN

The proof uses the following Lyapunov function (bounded, monotone decreasing pseudo energy function) of the Time-Discrete CNN:

$$V(Y(n)) = -\sum_{\nu=0}^{M\cdot N} Y_\nu(n)\left(\frac{1}{2}\sum_{\mu=0}^{M\cdot N}\mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N}\mathcal{B}(\nu,\mu)U_\mu + D_\nu\right) + \sum_{\nu=0}^{M\cdot N}\int_0^{Y_\nu(n)} g(y)dy \tag{14}$$

with

$$g(y) = \begin{cases} -1 & \text{for} \quad y=-1 \\ f^{-1}(y) = x & \text{for} \quad |y|<1 \\ 1 & \text{for} \quad y=1 \end{cases} \tag{15}$$

The boundedness of $V(Y(n))$ is shown in the same way as in [4]. The proof of the monotonicity deals with the first difference of $V(Y(n))$:

$$\begin{aligned}\Delta V(Y(n)) \quad = \quad & -\sum_{\nu=0}^{M\cdot N}\Delta Y_\nu(n)\left(\frac{1}{2}\sum_{\mu=0}^{M\cdot N}\mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N}\mathcal{B}(\nu,\mu)U_\mu + D_\nu\right) \\ & -\frac{1}{2}\sum_{\nu=0}^{M\cdot N}\sum_{\mu=0}^{M\cdot N}\mathcal{A}(\nu,\mu)Y_\nu(n)\ \Delta Y_\mu(n) \\ & -\frac{1}{2}\sum_{\nu=0}^{M\cdot N}\sum_{\mu=0}^{M\cdot N}\mathcal{A}(\nu,\mu)\,\Delta Y_\nu(n)\ \Delta Y_\mu(n) + \sum_{\nu=0}^{M\cdot N}\int_{Y_\nu(n)}^{Y_\nu(n+1)} g(y)dy\end{aligned}$$

According to the symmetry of $\mathcal{A}(\nu,\mu)$ and the mean-value theorem of the integration this results in:

$$\begin{aligned}\Delta V(Y(n)) \quad = \quad & -\sum_{\nu=0}^{M\cdot N}\Delta Y_\nu(n)\left(\sum_{\mu=0}^{M\cdot N}\mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N}\mathcal{B}(\nu,\mu)U_\mu + D_\nu\right) \\ & -\frac{1}{2}\sum_{\nu=0}^{M\cdot N}\sum_{\mu=0}^{M\cdot N}\mathcal{A}(\nu,\mu)\,\Delta Y_\nu(n)\ \Delta Y_\mu(n) + \sum_{\nu=0}^{M\cdot N}\Big(Y_\nu(n+1) - Y_\nu(n)\Big)g(\xi_\nu)\end{aligned}$$

with $\quad \xi_\nu = Y_\nu(n) + \gamma_\nu\,\Delta Y_\nu(n) \quad$ and $\quad \gamma_\nu \in [0,1]$.

Therefore:

$$\begin{aligned}\Delta V(Y(n)) \quad = \quad & -\sum_{\nu=0}^{M\cdot N}\Delta Y_\nu(n)\left(\sum_{\mu=0}^{M\cdot N}\mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N}\mathcal{B}(\nu,\mu)U_\mu + D_\nu - g(\xi_\nu)\right) \\ & -\frac{1}{2}\sum_{\nu=0}^{M\cdot N}\sum_{\mu=0}^{M\cdot N}\mathcal{A}(\nu,\mu)\,\Delta Y_\nu(n)\ \Delta Y_\mu(n)\end{aligned} \tag{16}$$

The proof of the monotonicity of $V(Y(n))$ is based on the following theorem which will be derived in the appendix:

Theorem 1 *If $\mathcal{A}(\nu,\mu)$ is a symmetric, positive semidefinite matrix then the following inequality is valid for $\xi_\nu = Y_\nu(n) + \gamma_\nu \triangle Y_\nu$ with $\gamma_\nu \in [0,1]$ and for all $n \geq 0$:*

$$\begin{aligned} &- \sum_{\nu=0}^{M\cdot N} \triangle Y_\nu(n) \left(\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu) Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu) U_\mu + D_\nu - g(\xi_\nu) \right) \\ \leq\ & \frac{1}{2} \sum_{\nu=0}^{M\cdot N} \sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu) \triangle Y_\nu(n) \triangle Y_\mu(n) \end{aligned} \tag{17}$$

where the identity is valid if and only if $\triangle Y_\nu(n) = 0$ for all ν.

Inserting equation (17) into equation (16) results in:

$$\triangle V(Y(n)) \leq 0 \tag{18}$$

which means that $V(Y(n))$ is monotone decreasing. The identity is valid if and only if $\triangle Y_\nu(n) = Y_\nu(n+1) - Y_\nu(n) = 0$ for all ν.

Every bounded, monotone decreasing sequence is convergent, i.e.

$$\lim_{n\to\infty} V(Y(n)) = \text{const.} \tag{19}$$

and therefore

$$\lim_{n\to\infty} \triangle V(Y(n)) = 0 \tag{20}$$

According to equation (18) $\triangle V(Y(n)) = 0$ corresponds to $Y_\nu(n+1) = Y_\nu(n)$ for all ν and all n which means that the Time-Discrete CNN converges towards an equilibrium.

5 Positive semidefiniteness of $\mathcal{A}(\nu,\mu)$

The proof in paragraph 4 requires that $\mathcal{A}(\nu,\mu)$ is positive semidefinite. A sufficient but not necessary condition for this can be derived from the theorem of Gerschgorin:

$$\mathcal{A}(\nu,\nu) \geq \sum_{\substack{\mu=0\\ \mu\neq\nu}}^{M\cdot N} |\mathcal{A}(\nu,\mu)| \quad \text{for all} \quad \nu = 0,1,\cdots,M\cdot N \tag{21}$$

Because of equation (10) this results in the following restriction for the original feedback operator values:

$$A(i,j;i,j) \geq \sum_{\substack{k=0\\ k\neq i}}^{M-1} \sum_{\substack{l=0\\ l\neq j}}^{N-1} |A(i,j;k,l)| \quad \text{for all} \quad i = 0,1,\cdots,M-1;\ j = 0,1,\cdots,N-1 \tag{22}$$

This restriction is fullfilled by most of the known feedback operators (e.g. noise removal, edge and corner detection, motion detection).

A Appendix: Proof of Theorem 1

The proof is based on the following theorem:

Theorem 2 $\triangle Y_\nu(n) = Y_\nu(n+1) - Y_\nu(n)$ *has the same sign as*

$$\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu)U_\mu + D_\nu - g(\xi_\nu)$$

i.e. the following equation is valid for all $\nu = 0, 1, \cdots, M \cdot N$:

$$\begin{aligned}\triangle Y_\nu(n) &= Y_\nu(n+1) - Y_\nu(n) = \alpha_\nu \left(\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu)U_\mu + D_\nu - g(\xi_\nu) \right) \\ & \quad \textit{with} \quad \alpha_\nu \geq 0\end{aligned}$$

Proof of Theorem 2 :

Inserting the state equation (2) and the output equation (3) into the first difference $\triangle Y_\nu(n)$ results in:

$$\begin{aligned}\triangle Y_\nu(n) &= f\left(\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu)U_\mu + D_\nu \right) - Y_\nu(n) \\ &= f\left(\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu)U_\mu + D_\nu + g(\xi_\nu) - g(\xi_\nu) \right) - Y_\nu(n)\end{aligned}$$

The Taylor's expansion of $f(\bullet)$ at $g(\xi_\nu)$ is given by:

$$\triangle Y_\nu(n) = f\left(g(\xi_\nu)\right) - Y_\nu(n) + \left(\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu)U_\mu + D_\nu - g(\xi_\nu) \right) \left.\frac{df(x_\nu)}{dx_\nu}\right|_{\dots}$$

Consequently with $f\left(g(\xi_\nu)\right) = \xi_\nu = Y_\nu(n) + \gamma_\nu \triangle Y_\nu(n)$:

$$(1-\gamma_\nu)\triangle Y_\nu(n) = Y_\nu(n) - Y_\nu(n) + \left(\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu)U_\mu + D_\nu - g(\xi_\nu) \right) \left.\frac{df(x_\nu)}{dx_\nu}\right|_{\dots}$$

Therefore with $\alpha_\nu = 1/(1-\gamma_\nu) \cdot df/dx \geq 0$:

$$\triangle Y_\nu(n) = \left(\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu)Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu)U_\mu + D_\nu - g(\xi_\nu) \right) \alpha_\nu \qquad \text{q.e.d.}$$

Proof of Theorem 1:

Case 1: $Y_\nu(n+1) = Y_\nu(n)$ i.e. $\Delta Y_\nu(n) = 0$ for all ν:
Both sides of the inequality (17) are zero and therefore identical.

Case 2: $Y_\nu(n+1) \neq Y_\nu(n)$ i.e. $\Delta Y_\nu(n) \neq 0$ for at least one ν:
Therefore according to theorem 2:

$$\alpha_\nu \neq 0 \quad \text{and} \quad \sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu) Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu) U_\mu + D_\nu - g(\xi_\nu) \neq 0$$

Inserting this into the inequality (17) results in:

$$\begin{aligned} &-\sum_{\nu=0}^{M\cdot N} \Delta Y_\nu(n) \left(\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu) Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu) U_\mu + D_\nu - g(\xi_\nu) \right) \\ &= -\sum_{\nu=0}^{M\cdot N} \alpha_\nu \left(\sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu) Y_\mu(n) + \sum_{\mu=0}^{M\cdot N} \mathcal{B}(\nu,\mu) U_\mu + D_\nu - g(\xi_\nu) \right)^2 \\ &< 0 \leq \frac{1}{2} \sum_{\nu=0}^{M\cdot N} \sum_{\mu=0}^{M\cdot N} \mathcal{A}(\nu,\mu) \triangle Y_\nu(n) \triangle Y_\mu(n) \qquad \textbf{q.e.d.} \end{aligned}$$

the last inequality is valid because $\mathcal{A}(\nu,\mu)$ is positive semidefinite

References

[1] Hubert Harrer, Josef A. Nossek, "On the Convergence of Time-Discrete Cellular Neural Networks", *Report No.: TUM-LNS-TR-90-14*, Technische Universität München, December 1990

[2] Hubert Harrer, Zbigniew Galias, Josef A. Nossek, "On the Convergence of Discrete-Time Neural Networks", *Report No.: TUM-LNS-TR-91-23*, Technische Universität München, November 1991

[3] Stefen Hui, Stanislaw H. Zak, "Dynamical Analysis of the Brain-State-in-a-Box (BSB) Neural Models",*IEEE Trans. on Neural Networks*, **Vol. 3**, January 1992

[4] L.O. Chua, Lin Yang, "Cellular Neural Networks: Theory", *IEEE Trans. on Circuits and Systems*,**CAS-35**, 1257-1272 (1988)

COUNTING STABLE EQUILIBRIA OF CELLULAR NEURAL NETWORKS - A GRAPH THEORETIC APPROACH

Paweł Kałużny

Nencki Institute, Pasteur 3, 02-093 Warsaw, Poland
fax ++48/22/22.53.42

Abstract. The paper presents a graph-theoretic method for computation of the number of stable equilibrium points and analysis of the structure of CNN equilibrium set. The considerations are based on one dimensional CNN layout. The total number of these equilibria, referred to as "output capacity" is shown to be equal to the number of paths in a graph derived from neighborhood consistency conditions. It may vary vastly depending on interaction weights and bias current of the cells. It is noted that output capacity equal to zero implies oscillations of the CNN state.

Introduction

The number of stable equilibrium points is one of major determinants of the signal processing capabilities of CNN. It gives the upper limit of the number of possible different outputs produced by the network, i.e. maximum number of patterns that can be retrieved, and is henceforth termed also "CNN output capacity". The structure of the set of equilibria is also practically very meaningful and puts constraints on the classes of patterns that can and can not be expected to emerge as the result of network computations. In the following, the algorithm is presented for computation of the number of different output vectors and used to study their dependency on some parameters of the network.

Notation. The cellular neural network considered is composed of N linearly ordered neurons, each governed by the state equation

$$C\frac{dx_i}{dt} = -\frac{1}{r}x_i + \sum_{j\in N(i)} A_j f(x_j) + J_i \ , \quad |x_i(0)|\leq 1 \ , \quad i=1..N \tag{1}$$

in standard notation [1,2]; f is identity function saturating at ± 1; J_i is bias current; r,C positive constants; vector $A=[A_{-k},..,A_0,..A_k]$ denote interaction mask in neighborhood of size k. For brevity we write x,y instead of standard V_x, V_y. Under relatively mild condition ($A_0 > 1/r$) [1,3], state trajectories converge to stable equilibria, given by

$$x_i = r\sum_{j\in N(i)} A_j f(x_j) + U \ , \quad |x_i|\geq 1 \ , \quad i=1..N. \tag{2}$$

where $U=rJ$ and tentatively assumed $J_i=J=const$. Any equilibrium satisfying $|x_i|>1$ for every cell $i=1..N$ is stable. In signal processing setting CNN transforms the initial state $x(t=0)$ (input) to one of its stable equilibria $x(t=\infty)$ and produce stable binary output vector $y=f(x)$. The actual transformation implemented depends upon the selection of masks A and bias currents J (which may include the contribution of masks B [1]). It was assumed that $r, C=1$ and neighbourhood radius $k=1$ in presented examples.

The algorithm of "output capacity" calculations

In order to find all stable equilibrium points of the CNN for given mask and bias current, we first determine all possible, stable neighborhoods for every cell, and then concatenate them to obtain full state vector.

In the first step, consider the state x of a typical cell and its neighborhood. Let $z_i = sign(x_i)$ and $z=[z_{-k},..z_0,..z_k]$ denote a binary pattern of output states in the neighborhood of a neuron spanned by mask A, positive states corresponding to $z_i=+1$ and negative to $z_i=-1$. There exist 2^{2k+1} different patterns z, and each of them can be conveniently represented as a unique binary number (-1 stands for binary digit 0). We mark by z^+, z^- respectively center-on ($z_0=+1$) and center-off ($z_0=-1$) neighborhood patterns. For example, if neighborhood radius $k=1$, eight different cases of z can be enumerated :

four center-off

0=[-,-,-]   1=[-,-,+]   4=[+,-,-]   5=[+,-,+]

and four center-on

2=[-,+,-]   3=[-,+,+]   6=[+,+,-]   7=[+,+,+].

The equilibrium conditions (2) allow only of some of them to exist in a stable output. To determine those configurations, termed "admissible neighborhood patterns", we write local equilibrium conditions for a cell and surrounding neighborhood as an alternative

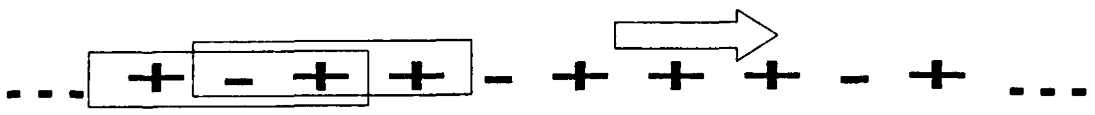
$$E(z^+) > 1-U \qquad or \qquad E(z^-) < -1-U \tag{3}$$

where the quantity

$$E(z) = r\sum_{j\in N} z_j A_j \tag{4}$$

represents "excitation level" of the center neuron, received from its neighborhood, given neighborhood pattern z.

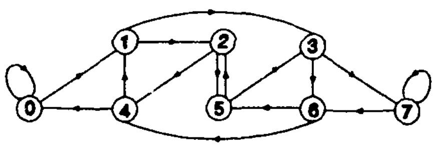
In the second step of the algorithm we consider the total number of N dimensional CNN output vectors that can be composed of $(2k+1)$ dimensional patterns z satisfying conditions (3). Conceptually, any such vector can be built by putting a number of admissible neighborhoods together in the string, starting from leftmost boundary cell to the right. If the network is spatially homogenous i.e. parameters A, U are identical for all the cells, the sets of admissible neighborhoods are also identical; otherwise admissible neighboroods have to be recomputed for every cell. For 1-neighborhoods the process is illustrated below



The pattern 5=[+-+] can be followed by 3=[-++] (as showed) or alternatively by 2=[-+-], but not by any other.

In general, the sequential process have to fulfill the constraining requirement that the patterns around two neighboring center cells should be identical except the left boundary cell of the preceding and the right of the following one.

This constraint can be expressed in the form of the neighborhood matching graph (G). The nodes of G correspond to admissible neighborhood patterns z and are marked with appropriate binary numbers. Two nodes i,j are connected by a directed branch from i to j if pattern i can be followed (on the right side) by j. The full neighborhood matching graph and its nodal incidence matrix M for 1-neighborhood is presented on Fig. 1. Note that it is independent on network parameters, except of neighbourhood radius, and its branching structure results from the properties of binary numbers. If some nodes are non admissible due to stability requirements (3) their corresponding rows and columns are set to zeros.



$$\mathbf{M} = \begin{array}{c|cccccccc|} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 4 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

Fig. 1 Full neighborhood matching graph for neighborhood of radius $k=1$ and its incidence matrix.

The central point of proposed algorithm is that the number of stable configurations of a segment of one dimensional CNN, composed of N cells, is equal to number of paths of length N-1 in the graph G (with non admissible nodes deleted). By a graph theoretic assertion (see e.g. [4], Theorem 9.10) this number, K_c, is the sum of all elements of nodal incidence matrix M of graph G, taken in $(N\text{-}1)$th power. Writing $K=M^N$ we obtain

$$K_c = \sum_{i,j} K_{ij} = \sum_{i,j} (M^N)_{ij} \tag{5}$$

The number of equilibrium states with given boundary neighborhoods i,j (possibly identical) is equal to $K_{ij}=(M^N)_{ij}$.

The above holds if the sets of admissible neighborhoods are identical for all the cells; if this is not the case, then to each cell corresponds the separate set of admissible nodes and the expression for matrix **K** is the product of appropriate reduced matrices M_i

$$K=M_2 * M_3 * \ldots * M_N \tag{6}$$

where each M_i preserves only branches from nodes admissible for the $(i\text{-}1)$th cell to nodes admissible for the ith cell; all other entries in full M are set to zeros.

Numerical examples and some implications for signal processing

Using the above algorithm the values of K_c were computed for selected cases of networks with neighbourhood radius $k=1$, with $A_0=2$, $N=10$ and A_{-1}, A_1 varying in the range (-2,2). The output capacity of such a family of networks (Fig. 2) may assume one of fixed values (4,24,466,4096) and thus the numbers of CNN equilibria may differ by several orders of magnitude depending on the values of interaction parameters A. The maximum of output capacity is attained for both interaction weights close to zero (middle part of the picture); the second largest value (466) is attained for approximately symmetric weights. With increasing N one observes combinatorial explosion of capacity due to multiple loops in G.

The actual input - output transformation implemented by CNN depends upon interaction mask A and bias U. With $U_i=U$ and fixed A, parameter U controls the number of stable equilibria and their domains of attraction, thus changing the structure and number of possible outcomes of network computations and possibility to distinguish between different inputs. As demonstrated by numerous simulations [2], the same mask may give different outputs depending on the bias U. Considering possible paths in the graph G allows for analysis of this influence and prediction of some aspects of the network behavior without

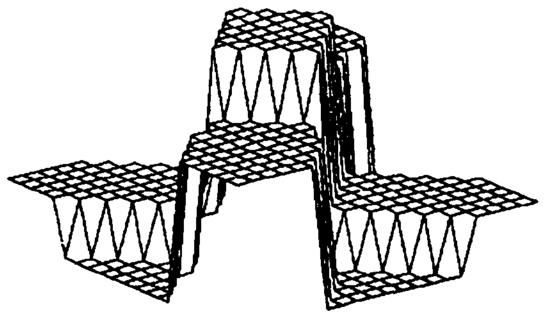


Fig. 2 $\mathrm{Log}_2(K_c)$ (vertically) versus weights A_{-1},A_1 $\in$(-2,2)(horizontal plane);$A_0=2$;$U=0$;$N=10$;view from above (0,0).

time consuming simulations and tests. As an example consider the mask $A=[1,2,1]$. The admissible nodes and numbers of equilibria K_c are presented in Tab. 1.

Table I Capacity and admissible neighborhoods for $A=[1,2,1]$.

U	admissible neighborhoods	capacity K_c $N=10/N=20$
$(-\infty,-3)$	{0,1,4,5}	4/4
(-3,-1)	{0,1,2,4,5,7}	5/5
(-1, 1)	{0,1,4,3,6,7}	466/57314
(1, 3)	{0,2,3,6,7}	5/5
$(3, \infty)$	{2,3,6,7}	4/4

On the base of analogy with classical FIR filters, it is expected to result in state $x(\infty)$ being a low pass filtered and thresholded variant of input i.e. the states of neighboring cells tend to be of the same sign, smoothing out minor discontinuities and noise in the input. One sees that this can take place only for U in (-1,1) because only there both "flat" patterns 0=[---] and 7=[+++] are admissible and connected. Otherwise , every input $x(0)$ gives either extremely ragged or saturated, flat output. Note that although both 0 and 7 are admissible also for U in (-3,-1) or (1,3) there is no path between them so they can not coexist in single state vector (comp. Fig. 1). The state space is partitioned onto K_c different domains, each leading to a different output.

Generally, it may be expected that if the application puts some constraints on the structure of desired outputs, in the similar manner the range of U can be inferred from the analysis of the graph of admissible neighborhoods and resulting capacities.

The mapping of input into an equilibrium fails if output capacity equals to zero. In fact, $K_c=0$ implies oscillations, as the CNN state trajectory is bounded and there are no stable equilibria (the algorithm counts critical points of (1) with all cell states outside unity, but $A_0>1/r$ assures instability of any others). This situation usually arises due to constraints introduced by network boundary cells, splitting the admissible nodes into non-connected subsets.

References

[1] L. Chua, L. Yang, Cellular Neural Networks : Theory and Applications, IEEE Trans., CAS-35, 10, 1988.

[2] Proceedings of International Workshop On Cellular Neuronal Networks and Their Applications, Budapest, 1990.

[3] L. Chua and C.W. Wu, On the Universe of Stable Cellular Neural Networks, Memo UCB/ERL M91/31.

[4] N. Deo, Graph Theory. With Applications to Engineering and Computer Science. Prentice-Hall, 1974.

Associative Memory Design Using Space-Varying Cellular Neural Networks

G. Martinelli and R. Perfetti

University of Rome "La Sapienza", Info-Com Dept.
via Eudossiana, 18 - 00184 Rome, Italy
Fax +39 6 4873300

Abstract

Space-varying Cellular Neural Networks (CNN's) are characterized by nearest-neighbor interconnections represented by a cell-dependent connectivity matrix A_{ij}. This biologically-inspired architecture has many potential applications. In the paper an associative memory design is presented.

I. INTRODUCTION

Cellular Neural Networks represent a powerful approach to parallel, nonlinear signal processing, that can be exploited for accomplishing various types of computational tasks. The simple architecture, with local connectivity, and the analog nature of computation make it feasible the VLSI realization of large networks. The basic model introduced by Chua and Yang [1,2] is characterized by linear interactions among the different cells. The interactions are represented by a space-invariant and time-invariant feedback operator, therefore called "cloning template". Several generalizations of this basic model have been described in the recent literature. In [3] CNN's with nonlinear and delay-type template elements have been introduced. In [4] an adaptive cloning template is used to perform a signal processing task. In the present paper, the space-invariant property is removed. The resulting CNN, named *space-varying* in the following, is characterized by nearest-neighbor interconnections represented by a cell-dependent connectivity matrix A_{ij} . This assumption, which is a reasonable one for biological neural networks, leads to an extra complication in the VLSI realization. However, the space-varying condition extends the applicability of CNN's to a wide variety of problems. In this paper space-varying CNN's are used to design a cellular associative memory. In an associative (or content-addressable) memory (AM) the information is retrieved from a corrupted or incomplete version of it used as a key. Associative memories can be realized by using dynamical neural networks [5]. A stored pattern corresponds to an equilibrium state of the network: the dynamic behaviour, starting from states sufficiently close (in the Hamming sense) to the equilibrium point, will converge to it, giving an error correction capability. Several design examples have been reported that make use of full-connection neural networks (see [6] for a survey). The design of CNN's to function as associative memories has been addressed in a previous work [7]. However, the approach followed in [7] is based on the classical Hebb rule. It is well-known that a full-connection Hebbian associative memory has a very low capacity, unless thc patterns to be stored are mutually orthogonal. Since it is reasonable that the memory capacity decreases as the number of interconnections decreases, a different design method must be found. In the following, the learning algorithm proposed by Zou et al. [8] is applied to compute the connectivity matrices A_{ij}, i =1,..., M, j =1,..., N of an MxN CNN used as associative memory. The details of the model and of the learning algorithm are described below.

II. CNN MODEL

With reference to [1], the normalized CNN equations are

$$\frac{dv_{x\,ij}}{dt} = -v_{x\,ij} + A_{ij} * v_y \qquad i=1,\dots, M,\ j=1,\dots, N \tag{1}$$

where $v_{x\,ij}$ represents the state variable of cell C(i, j). The compact notation $A_{ij}*v_y$ represents a spatial convolution and is defined as follows:

$$A_{ij} * v_y = \sum_{C(k,l)\in N_r(i,j)} A_{ij,kl}\, v_{y\,kl} \tag{2}$$

A_{ij} is the feedback operator, C(k,l) denotes the cell on the kth row and lth column, in a rectangular array. $N_r(i,j)$ represents the r-neighborhood of cell C(i,j); a value r =1 will be assumed throughout the paper. v_{yij} is the output variable. Note that the control operator **B** and the threshold I are not used. The input-output transfer characteristic of a cell is: $v_{y\,ij} = 0.5\,(|v_{x\,ij} + 1| - |v_{x\,ij} - 1|)$.

The symmetry assumption is made:

$$A_{ij,kl} = A_{kl,ij} \tag{3}$$

for every i, j, k, and l.

Condition (3) guarantees the complete stability, as shown in [1]. It is well known that, if $A_{ij\,ij} > 1$, the only observable d.c. solutions correspond to binary outputs, i.e. $v_{yij} \to +1$ or -1 , for all i and j, as $t \rightarrow \infty$. Moreover, it can be shown [8] that an asymptotically stable, bipolar solution $v_{yij} \in \{-1, +1\}$, i =1,..., M, j =1,..., N, exists iff it is

$$A_{ij} * v_y - 1 \geq 0 \quad \text{if } v_{yij} = +1 \qquad (4a)$$

$$- A_{ij} * v_y - 1 \geq 0 \quad \text{if } v_{yij} = -1 \qquad (4b)$$

for i =1,..., M, j =1,..., N.

Conditions (4) are nece~~ss~~ary and sufficient. So they can be used either to check if a bipolar vector is a network solution or to determine the network parameters in order to have a desired solution. In the following, they will be used as a design tool.

III. ASSOCIATIVE MEMORY DESIGN USING LINEAR RELAXATION

The design of an associative memory based on a space-varying CNN consists in the determination of the connectivity matrices $A_{ij} \in R^{3x3}$, i =1,..., M, j =1,..., N, so as to satisfy the following requirements:

(i) System (1) will have no periodic or chaotic solutions.

(ii) A desired set of matrices $v_y^{(k)} \in \{-1, +1\}^{MxN}$, k=1,..., P, correspond to as many asymptotically stable states of the network.

The symmetry condition guarantees point (i). Point (ii) is satisfied by imposing the constraints (3) for the desired P patterns, assuming they are consistent. This leads to a set of linear constraints that can be solved by several efficient methods, such as numerical relaxation or linear programming. The method we followed is based on the linear relaxation algorithm [8]. Taking into account the symmetry property, a link between two adjacent cells is characterized by only one strenght value. Assuming the self-connection $A_{ij,ij}$ is chosen *a priori*, the number of different parameters to be computed is easily determined: it is L = (N-1) M + (M-1) N+ 2 (N-1) (M-1). Hence, PxMxN simultaneous linear constraints must be satisfied in order to obtain the L unknowns. Let S be the solution space.

Let us introduce the following formalism:

A_0 is the common self-feedback value (the only restriction is $A_0 > 1$);

$X=[\, A_{11,12}\ A_{11,21}\ A_{11,22} \ldots\ldots\ldots \,] \in R^L$ is the vector of unknowns;

$X_{ij} \in R^8$, i =1,...,M, j =1,..., N, is a subvector containing only the unknowns relative to the cell C(i,j):

$$X_{ij} = [A_{ij;\, i-1,j-1}\ A_{ij;\, i-1,j}\ A_{ij;\, i-1,j+1}\ A_{ij;\, i,j-1}\ A_{ij;\, i,j+1}\ A_{ij;\, i+1,j-1}\ A_{ij;\, i+1,j}\ A_{ij;\, i+1,j+1}\,]^T$$

$Y_{ij}^{(k)} \in \{-1,+1\}^8$, i =1,...,M, j =1,..., N, k =1,..., P, is a vector containing only the output variables relative to the cell C(i,j) and to the kth pattern to be stored, except for the term $v_{y\,ij}$.:

$$Y_{ij}^{(k)} = [\, v_{y\,i-1,j-1}^{(k)}\; v_{y\,i-1,j}^{(k)}\; v_{y\,i-1,j+1}^{(k)}\; v_{y\,i,j-1}^{(k)}\; v_{y\,i,j+1}^{(k)}\; v_{y\,i+1,j-1}^{(k)}\; v_{y\,i+1,j}^{(k)}\; v_{y\,i+1,j+1}^{(k)}\,]^T$$

The learning algorithm is as follows:

```
Step 0. Choose $X \in R^L$ arbitrarily.

Step 1. For k =1 to P
        For i =1 to M
        For j =1 to N
        if ($v_{y\,ij}$ =1) and ($X_{ij}^T Y_{ij}^{(k)} + A_0 - 1 < 0$)
        then
            $X_{ij} = X_{ij} - 2 \frac{X_{ij}^T Y_{ij}^{(k)} + A_0 - 1}{\|Y_{ij}^{(k)}\|^2} Y_{ij}^{(k)}$
        else
        if ($v_{y\,ij}$ = - 1) and ($- X_{ij}^T Y_{ij}^{(k)} + A_0 - 1 < 0$)
        then
            $X_{ij} = X_{ij} - 2 \frac{X_{ij}^T Y_{ij}^{(k)} - A_0 + 1}{\|Y_{ij}^{(k)}\|^2} Y_{ij}^{(k)}$

Step 2. if $X \in S$ then stop
        else goto Step 1.
```

IV. SIMULATION RESULTS

In Fig. 1 a simulation example is shown. The rightmost images represent the six patterns to be stored in the memory realized by a 6x6 CNN. The self-feedback is A_0 =2. All the unknowns are initially equal to 2. The algorithm described above was utilized to compute the L=110 different connection values. The algorithm converged after 105 iterations. At this point the network dynamics was simulated by numerical integration of Eqns. (1). The leftmost images in Fig. 1 represent corrupted versions of the stored images (on the right). Using these noisy images as initial states, the desired output image is obtained as the final state of the network. Even if the considered example is characterized by modest dimensions (36 cells), the simulation results are very encouraging. In fact, the ratio of the number of stored patterns with respect to the number of cells, equal to 0.167, and the ratio of the number of stored patterns with respect to the number of interconnections, equal to 0.027, are quite satisfactory if compared to those obtained using full-connected neural networks. Note that the present method guarantees that the given set of patterns are all stable states of the CNN, if S is nonempty.

References

[1] L.O. Chua and L. Yang, "Cellular neural networks: Theory", IEEE Trans. Circuits Syst., vol. CAS-35, pp. 1257-1272, October 1988

[2] L.O. Chua and L. Yang, "Cellular neural networks: Applications", IEEE Trans. Circuits Syst., vol. CAS-35, pp. 1273-1290, October 1988

[3] T. Roska, L.O. Chua, "Cellular neural networks with nonlinear and delay-type template elements", Proc. First IEEE Int. Workshop on Cellular Neural Networks Appl., CNNA-90 (Budapest), 1990, pp. 12-25

[4] R. Perfetti, "Cellular neural network for fast adaptive equalization", to be published on Int. Journal on Circuit Theory and Appl.

[5] R.J. McEliece, E.C. Posner, E.R. Rodemich and S.S. Venkatesh, "The capacity of the Hopfield associative memory", IEEE Trans. Inform. Theory, Vol. IT-33, July 1987, pp. 461-482

[6] A.N. Michel, J.A. Farrell, "Associative memory via artificial neural networks", IEEE Control Systems Magazine, April 1990, pp. 6-17

[7] S. Tan, J. Hao and J. Vandewalle, "Cellular neural networks as a model of associative memories", Proc. First IEEE Int. Workshop on Cellular Neural Networks Appl., CNNA-90 (Budapest), 1990, pp. 26-35

[8] F. Zou, S. Schwarz, J. A. Nossek, "Cellular neural network design using a learning algorithm", Proc. First IEEE Int. Workshop on Cellular Neural Networks Appl., CNNA-90 (Budapest), 1990, pp. 73-81

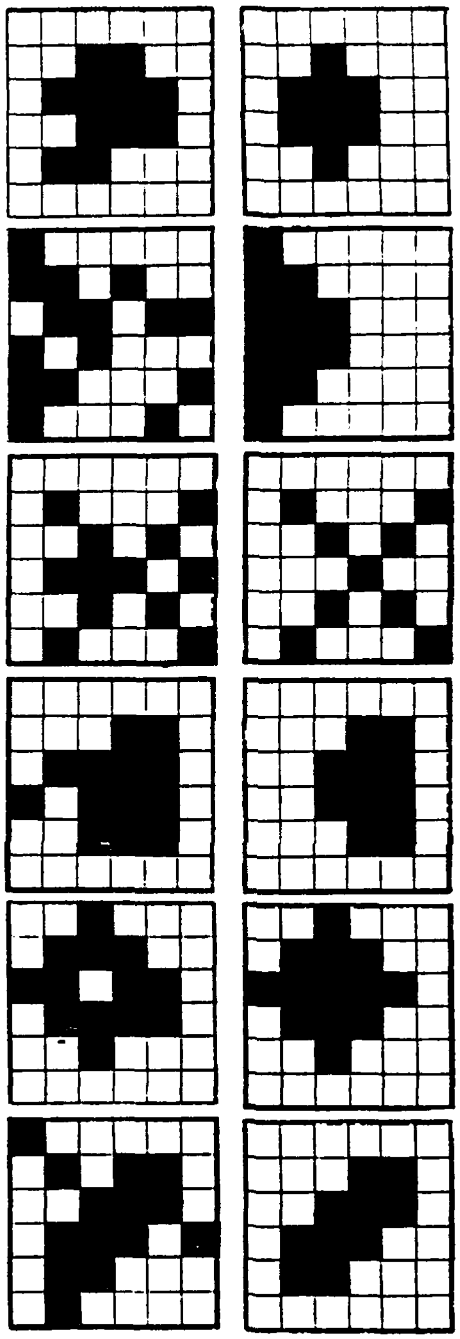


Fig. 1. Simulation example. The noisy images on the left are used as initial states of a 6x6 CNN. The corresponding final states, on the right, coincide with the images stored in the cellular associative memory.

Theoretical and Experimental Studies of Oscillations in Simple CNN Structures

Maciej J. Ogorzałek
Andrzej Dąbrowski
Department of Electrical Engineering
University of Mining and Metallurgy
al. Mickiewicza 30, 30-059 Kraków, Poland
tel. (+48 12) 33 91 00 ext. 3613; fax. (+48 12) 33 10 14
e-mail: geogorza@plkrcy11.bitnet

Abstract

In this paper we study possibilities of existence of oscillatory behavior in a simple closed loop linear interconnections of cells. We established analytical sufficient conditions for existence of oscillations in a ring of any number of interconnected cells. Simulation experiments show very good agreement with analytical predictions. Oscillations have been also confirmed in the cases of more complex interconnections of cells and in particular in the case of self-feedback existing in all cells.

1. Introduction

Dynamic behaviour of cellular neural networks is fairly well understood in cases of symmetric interconnections [6]. However it has been pointed out by many authors that even simplest neural structures can exhibit complex dynamics, typical for nonlinear systems, such as oscillations, fractal basins of attraction or chaotic behaviour [1-4], [7-8]. Even the simplest two- and three-cell interconnections are prone to exhibit oscillatory and chaotic behavior and a variety of bifurcation phenomena [8-9].

In this paper we study the possibilities of oscillatory behavior in simple cellular neural networks and in particular in closed loop interconnections of cells. Several simulation experiments and also laboratory tests have been carried out confirming the existence of oscillatory solutions in this kind of structures with nonsymmetric templates or non-uniform interconnections (ie. interconnection templates are not identical for all cells). It is possible to show using rigorous mathematical reasoning that an interconnection of at least three cells is likely to become oscillatory if the interconnection weights (or the cell gains) are appropriety adjusted. In particular it is possible to show that asymmetric templates with no self-feedback give rise to oscillations when applied in a ring structure of the network. Influence of the self-feedback is also studied in simulation experiments. Presented results constitute a contribution to qualitative analysis of nonsymmetric cellular neural networks and allow deeper insight into networks dynamics.

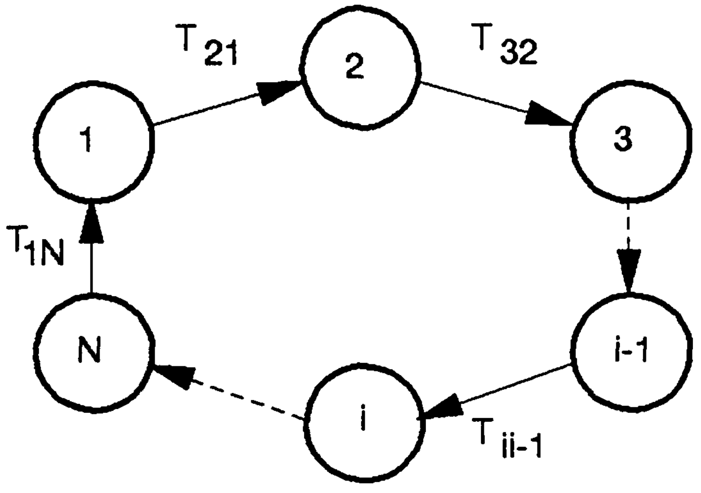


Figure 1. Considered interconnection of cells.

2. Oscillation criteria for closed-loop interconnection of cells

We assume that the dynamics of the i-th cell can be described by a first order differential equation of the form:

$$\frac{du_i(t)}{dt} = \frac{1}{C_i}[\Sigma_j T_{ij} f_j[u_j(t)] - \frac{u_i(t)}{R_i} + I_i] \tag{1}$$

where: $u_i(t)$ represents the voltage across the i-th storage capacitor, T_{ij} represent the interconnection weights (influece of neighboring cells), f_j- a saturation-type, monotonically increasing, continuous function.

Let us assume that the cells are interconnected in a one-dimensional array (Fig.1). This imposes a constraint on connectivity matrix $T = [T_{ij}]$. The equations (1) become $(i = 2, ...N)$:

$$\frac{du_1(t)}{dt} = \frac{1}{C_1}[T_{1N} y_N(t) - \frac{u_1(t)}{R_1} + I_1] \tag{2}$$

$$\frac{du_i(t)}{dt} = \frac{1}{C_i}[T_{ii-1} y_{i-1}(t) - \frac{u_i(t)}{R_i} + I_i] \tag{3}$$

These equations can be rewritten in the form:

$$\frac{du_1(t)}{dt} = F_1(u_N, u_1) \tag{4}$$

$$\frac{du_i(t)}{dt} = F_i(u_{i-1}, u_i) \tag{5}$$

These equations describe dynamic behavior of a particular type of cellular neural network - namely the A templates are row-vectors with 0 self-feedback (central element) and only one non-zero element (thus are nonsymmetrical).

The results presented in this study are based on an interesting existing result [5]:

THEOREM 1.

Let functions F_i satisfy :
$F_i(0,0) \geq 0$, $i = 1,...N$ and $F_1(u_N, 0) > 0$ for all $u_N \geq 0$
and the following inequalities hold on a closed subset $\overline{P} \in R^n$:
$\frac{\partial F_i}{\partial u_i} < 0$ and $\frac{\partial F_i}{\partial u_{i-1}} > 0$ for $2 \leq i \leq N$ and $\frac{\partial F_1}{\partial u_N} < 0$.
The system (5) has a unique steady state $u^* = [u_1^*, ..., u_N^*]^T$ in the set P, with $F_1(u_N, u_1) < 0$ if $u_N > u_N^*$ and $u_1 > u_1^*$, while $F_1(u_N, u_1) > 0$ if $u_N < u_N^*$ and $u_1 < u_1^*$. Also, $\frac{\partial F_1}{\partial u_1}$ is bounded above in $\overline{P}$. Let $J = F_u(u^*)$ - the Jacobian matrix of F at u^*. Suppose that J has no repeated eigenvalues.
Then:
if J has any eigenvalues with positive real parts, the equation (5) has a nonconstant periodic solution in the set P.

COROLLARY 1.

If for $N \geq 3$ all the weights T_{ij} are zero except $T_{1N}, ...T_{ii-1}$, $i = 2,...N$ and exactly one element among the nonzero ones is negative while all others are positive furthermore $\frac{T_{1N}}{2} + I_1 = 0$ and $\frac{T_{ii-1}}{2} + I_i = 0$ for $i = 2,...N$, then if the Jacobian matrix satisfies the condition as in Theorem 1 and the closed loop of neural cells possesses a nontrivial periodic solution.

COROLLARY 2.

Assuming that all cells have identical nonlinear saturation-type characteristics with g_0 - gain at the origin the condition imposed on the eigenvalues of the Jacobian matrix can be replaced by:

$$\frac{2}{(-T_{1N}T_{21}...T_{NN-1})^{\frac{1}{N}} cos(\frac{\pi}{N})} < g_0 \tag{6}$$

and in particular when $g_0 = 1$ as in the case of typical CNN cells:

$$\frac{2}{(-T_{1N}T_{21}...T_{NN-1})^{\frac{1}{N}} cos(\frac{\pi}{N})} < 1 \tag{7}$$

3. Experimental results

Extensive simulation and laboratory experiments have been carried out to investigate the dynamic behaviors in ring structures of cellular neural networks. In our experiments we used a modification of the op-amp cell model proposed by Chua and Yang [10]. Below we present some comparative results obtained in interconnections of 3, 6 and 9 neural cells. The time constant of the cells used was fixed $CR = 10^{-6}s$. Waveforms preceding the onset of oscillations and those showing how the oscillations are generated in the system are shown in Fig.2-4. It is interesting to notice that the greater the number of neurons in the ring the smaller is the weight value by which the oscillations are generated in the system. The frequency of oscillations diminishes with growing length of the ring and the amplitude of the state variables (the capacitor voltages) depend on the weight value (template coefficient) and is bounded as given by the formula (3) of [10] - the greater the

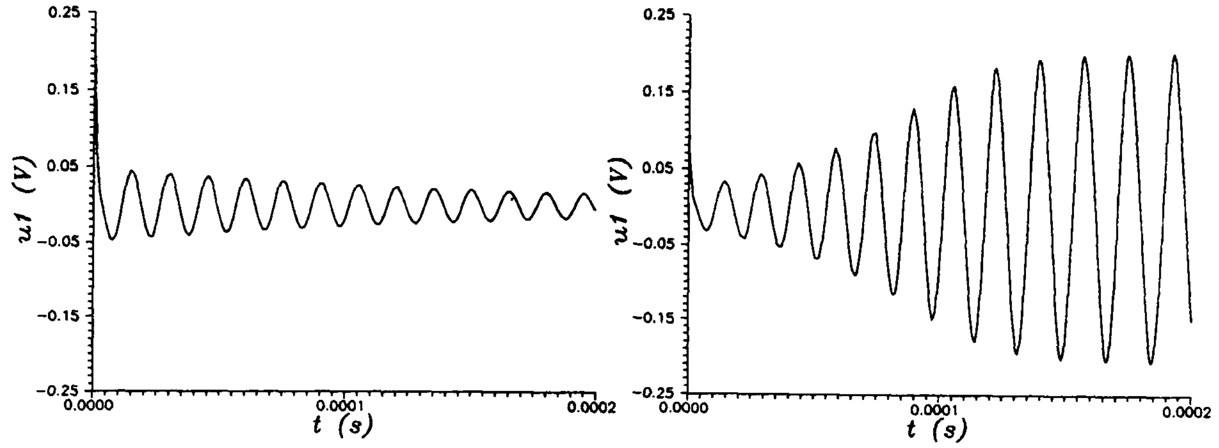


Figure 2. Typical waveforms observed in a 3-cell ring (N=3). $T_{ii-1} = 1.3mS$ (a) , $T_{ii-1} = 1.4mS$ (b).

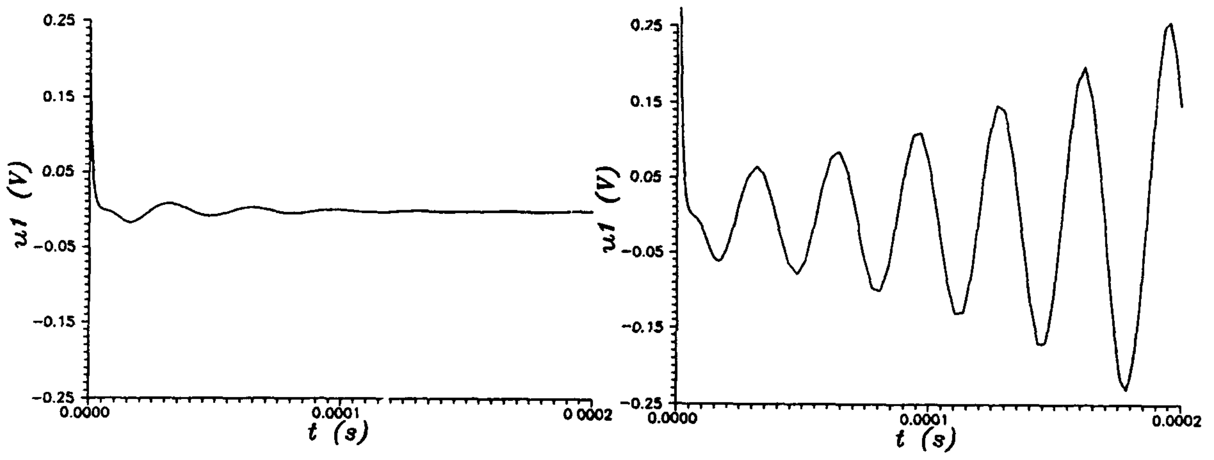


Figure 3. Typical waveforms observed in a 6-cell ring (N=6). $T_{ii-1} = 1.0mS$ (a) , $T_{ii-1} = 1.1mS$ (b).

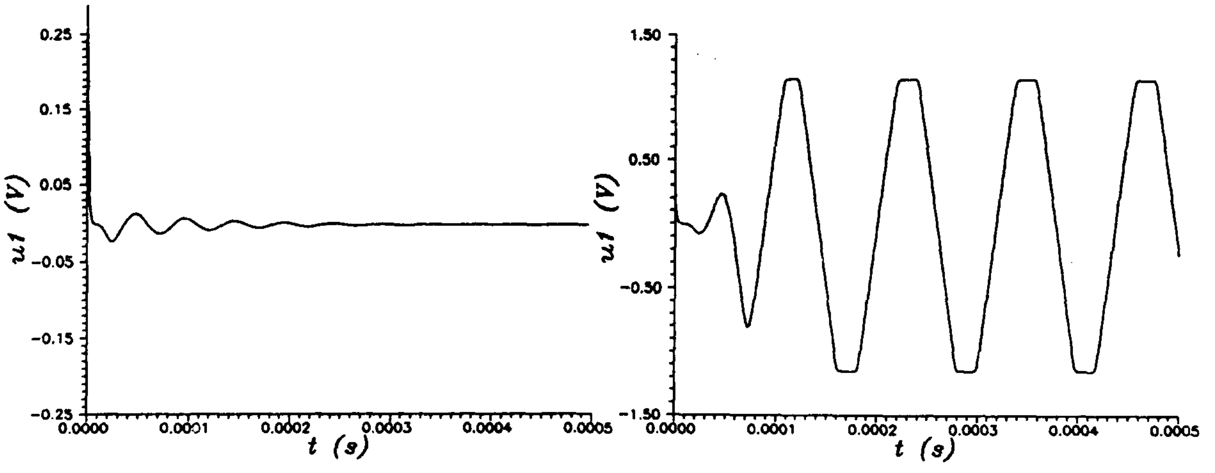


Figure 4. Typical waveforms observed in a 9-cell ring (N=9). $T_{ii-1} = 1.0mS$ (a) , $T_{ii-1} = 1.2mS$ (b).

weight value the greater the amplitude of the voltages. This observation might be crutial in the design as the upper bound of this amplitude is imposed by the dynamic range and voltage swing of the actual amplifier used.

In further experiments we tested the system behavior in the case when the self-feedback coefficients were non-zero. Introduction of a small self-feedback caused the frequency of oscillations diminish and the amplitude grow in comparison with the ring without self-feedbacks. The formula (3) of [10] for the amplitude bounds still holds.

In several experiments we observed that introduction of self-feedback caused the network oscillate even in the case where without the self-feedback the system was allways going towards an equilibrium point.

All experiments show substantial influence of the particular circuit implementation used on the performance of the system. Depending on the actual active elements used the amplitude of oscillation changes and distortion of the signal (clipping) is observed.

Further studies are concentrated on dynamic behaviors of different kinds of regular interconnections of neural cells and bifurcation mechanisms in large cellular neural networks.

4. Conclusions

- It follows from Corollary 2. that it is always possible to adjust the weights of cell interconnections (template coefficients) to obtain oscillations in the closed loop of three or more cells. It is possible to ensure oscillatory behavior in a ring of odd number of identical cells. In the case of even number of cells at least one of the cells must have a different template (sign of the weight).
- Similar result could be obtained provided the gains of the nonlinear characteristics could be adjusted.
- It is interesting to note that the assumption about the fixed gain in basic CNN definition is very important as two systems with exactly the same topology and connection matrices can have different dynamic behaviours depending only on the gain factors.
- Experiments fully confirm the theoretical reasoning given above. Morover several other types of oscillatory behavior have been confirmed. These include parasitic oscillations in certain realisations of neural cells.
- Possibilities of existence of oscillatory solutions in simple CNN structures (as considered above) may confirm the difficulties in design of CNN with templates having coefficients of both positive and negative signs.
- The circuits discussed in this paper are relatively simple however one should consider that similar effects are even more likely to occur in two- or three- dimensional interconnection structures. The undesirable effects of phase shifts as could be seen from our simple examples are likely to occure in the case of opposite sign templates and may lead to complex collective behaviour of the network including sustained oscillations or so-called ringing transients.

- Analysis of CNN from the point of view of nonlinear dynamics and nonlinear oscillations in particular should prove useful in solving important theoretical and practical problems and may lead to new areas of applications.

5. Acknowledgment

This research has been supported by Polish Committee of Scientific Research grant No. 3 0678 91 01.

6. References

1 T.Allen "On the arithmetic of phase locking: Coupled Neurons as a lattice on R^2". Physica 6D, pp.305-320, 1983.

2 L.O.Chua, T.Roska "Stability of a Class of Nonreciprocal Cellular Neural Networks". ERL Memo UCB/ERL M89/100, University of California Berkeley, 1989.

3 K.L.Babcock, R.M.Westervelt "Stability and Dynamics of Simple Electronic Neural Networks with Added Inertia". Physica 23D, pp.464-469, 1986.

4 B.Barranco, E.Sanchez-Sinencio, A.Rodriguez-Vazquez, J.Huertas "A Programmable Neural Oscillator Cell". IEEE Trans. Circuits Systems, vol.CAS-36, No.5, pp.756-761, 1989.

5 S.Hastings, J.Tyson, D.Webster "Existence of Periodic Solutions for Negative Feedback Cellular Control Systems". Journal of Differential Equations, vol.25, pp.39-64, 1977.

6 F.Zou, J.A.Nossek "Stability of Cellular Neural Networks with Opposite-sign Templates". Report TUM-LNS-TR-15, 1990, Technical University Munich.

7 F.Zou, J.A.Nossek "A chaotic attractor with cellular neural networks". IEEE Trans. Circuits Systems, vol.38, pp.811-812, 1991.

8 J.A.Nossek, F.Zou "Bifurcation and chaos in cellular neural networks". World Congress of Nonlinear Analysts, Tampa, August 1992 (in press).

9 T.Saito " Chaos from a Forced Neural-Type Oscillator". Trans. IEICE Japan, vol.E 73, No.6, pp. 836-841, 1990.

10 L.O.Chua, Lin Yang "Cellular neural networks: Theory". IEEE Trans. Circuits Systems, vol.35, pp.1257-1272, 1988.

GENERALIZED CNN: POTENTIALS OF A CNN WITH NON-UNIFORM WEIGHTS

Marco Balsi

Dipartimento di Ingegneria Elettronica
Universita' di Roma "La Sapienza"
via Eudossiana 18, Roma, Italy I-00184

Abstract - *A generalization of the Cellular Neural Network paradigm is obtained by removing the uniformity constraint on weight values. Such Generalized CNNs are capable of new tasks, such as function approximation or associative memory. A stability analysis of these networks is presented. Adaptation and application of a gradient descent learning algorithm is then discussed.*

Introduction - The most important characteristic of Cellular Neural Networks (CNN) as defined by Chua & Yang [CHU88] is locality of connections, that greatly simplifies the layout problem for IC realization.

The said model, however, involves another significant constraint in the uniformity of weight values, so that processing consists of a spatial convolution with an operator defined by the cloning template. In this way, and also with the extension to non-linear and delay-type templates [ROS90], many image processing problems have been successfully solved [CNNA90], always using the network with strong enough self-feedback as to obtain saturated (i.e. ±1) steady-state outputs.

If the constraint on uniformity of weights is removed, new applications may be conceived for CNNs, such as Content-Addressable Memory (CAM) [TAN90], classification, function approximation, that cannot be implemented by traditional CNNs.

I shall call Generalized CNN (GCNN) a network which is identical to Chua & Yang's [CHU88] when parameters $A(i,j;k,l)$, $B(i,j;k,l)$, $I(i,j)$ are allowed to vary arbitrarily while respecting the locality condition. GCNNs are included in the extended definition of CNN recently given by Roska [ROS92]. In the following, normalization $R_x = 1$, $C = 1$ is applied, and $B(i,j;k,l) = \delta_{ij}^{kl}$ without loss of generality. Voltages v_x, v_y, v_u in [CHU88] will be denoted x, y, u in the following. I shall always consider networks with clamped inputs and neglect specifying that sums over cell indices must be taken inside the significant neighborhood.

0-7803-0875-1/92 $3.00 ©1992IEEE

Therefore, GCNN neuron dynamics is written as follows:

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \Sigma_{kl}\ A(i,j;k,l)f(x_{kl}(t)) + I(i,j) + u_{ij} \qquad (1)$$

Due to the peculiarities of the model, development of CNN theory has followed methods that are quite different from those usually applied in Neural Network (NN) theory and sometimes are more similar to Digital Signal Processing methodologies. This is most evident concerning one of the most important topics of NN theory: learning.

In this paper I shall discuss stability of GCNNs, based on known results of CNN and NN theory. On this basis I discuss possible applications of the paradigm. The learning issue is later confronted, by considering the possibility of application of standard NN algorithms, especially a gradient descent method.

Stability - It is possible to apply to GCNNs many known results from CNN and NN theory.

Theorem 1 *(bound on state values)*: If $\forall\ i,j\ \ |x_{ij}(0)| \le 1,\ \ |u_{ij}| \le 1$ and $|I_{ij}| \le I$, then all states are bounded for all time $t > 0$ and the bound x_{max} is computed as follows:

$$x_{max} = 2 + I + \max_{i,j} \Sigma_{k,l}\ |A(i,j;k,l)|$$

Proof: It is theorem 1 of [CHU88].

Theorem 2 *(stability of reciprocal nets)*: If $\forall\ i,j,k,l\ \ A(i,j;k,l) = A(k,l;i,j)$, then the network is globally asymptotically stable.

Proof: Follows immediately from theorems 2,3 and 4 of [CHU88].

Theorem 3 *(saturated steady-state output)*: If $\forall\ i,j\ \ A(i,j;i,j) > 1$ then magnitude of stable states must be grater than 1.

Proof: It is theorem 5 of [CHU88].

Theorem 4 *(stability of low-level-feedback nets)*: If $\forall\ i,j$

$$A(i,j;i,j) + 1/2\ \Sigma_{k,l} \left(|A(i,j;k,l)|+|A(k,l;i,j)|\right) < 1$$

(where $A(i,j;i,j)$ may also take negative values) then the network is globally asymptotically stable.

Proof: Follows immediately from theorem 3 of [HIR89].

Theorem 5 *(stability of positive-cell-linking networks)*: If $\forall\ i,j,k,l$ $A(i,j;k,l) \ge 0$ and $\forall\ i,j,k,l$ there is a path on the network graph from i,j to k,l passing only through positive weights, then the network is almost everywhere asymptotically stable.

Proof: Follows from theorem 1 of [CHU90].

Theorem 6 *(stability of feed-forward processing GCNNs)*: If a GCNN can be decomposed into a cascade of globally asymptotically stable GCNNs, it is also globally asymptotically stable.

Proof: It is a corollary of theorem 5 of [HIR89].

Applications - Leaving away consideration of networks having periodic or chaotic attractors [BAR92], two modes of operation are interesting for neural computing: the first case is when, for every clamped input, the net is globally convergent [HIR89] regardless of initial conditions. In this case the net performs a classification of inputs when it has saturated final states, or otherwise it implements a continuous mapping from inputs to continuous-valued outputs. It need not be reset and can be cascaded by transferring the output of a stage to the input of the following one.

In the second mode inputs are clamped to a fixed value and the net has multiple attraction basins (it is convergent), depending on initial conditions. In this case classification or mapping is done between initial conditions and final outputs. This way of functioning is typical of Content-Addressable Memories (CAM).

In this paper I restrict consideration to networks operating in the first mode.

A task for such a network may be described in general by a function $\mathcal{F}: \mathcal{I} \to \mathcal{O}$ where input space $\mathcal{I}$ and output space $\mathcal{O}$ may be continuous or discrete. For instance, in a classification problem $\mathcal{I} \subseteq \mathbb{R}^n$ and $\mathcal{O} \subset \mathbb{Z}^m$ for suitable n and m. For this reason, I shall discuss learning problems as applied to function approximation tasks for boolean and continuous mappings.

Learning - All the work that is currently being done on CNN learning has been based on the fact that, unlike all other NN models, CNNs have very few parameters [HAR91] [SZO91]. This is not the case for GCNNs, and for this reason it is necessary to exploit classical learning algorithms from NN theory. The only example in literature may be found in [TAN90], where the Hebb rule was used for CAM purposes. Other possibilities include stochastic methods, such as simulated annealing, or gradient descent algorithms. In this paper I discuss application and modification of a gradient descent algorithm: Recurrent Back-Propagation (RBP) [PIN87], which is a generalization of the well known Back-Propagation to non-feed-forward networks.

Recurrent Back-Propagation - Consider GCNN dynamics (1). A sigmoidal output function $f(x) = 2(1+\exp(-\beta x))-1$ was chosen, because the usual piecewise linear output function, having zero derivative outside the interval (-1,1), would give more problems of local minima during learning.

Denote γ^{μ} the desired steady state output matrix with input matrix u^{μ}. In the general case, u and γ take significant values over some units only (input and output neurons). Choose error measure E as follows:

$$E = 1/2 \, \Sigma_{\mu}\Sigma_{ij} \left[E^{\mu}_{ij}\right]^2; \; E^{\mu}_{ij} = \alpha_{ij}[\gamma^{\mu}_{ij} - f(\underline{x}^{\mu}_{ij})]$$

where $\underline{x}^{\mu}$ is the stable state with input u^{μ}, α_{ij} is 1 for output neurons and 0 for neurons whose state is hidden to the external environment.

Gradient descent over error surface E yields the usual delta rule, which may be written as:

$$A_{k+1}(p,q;r,s) = A_k(p,q;r,s) + \Delta A_k(p,q;r,s); \; I_{k+1}(i,j) = I_k(i,j) + \Delta I_k(i,j)$$

$$\Delta A(p,q;r,s) = -\eta \frac{\partial E}{\partial A(p,q;r,s)} = \eta \Sigma_{\mu} t^{\mu}_{pq} f(x^{\mu}_{rs}); \; \Delta I(i,j) = \eta \Sigma_{\mu} E^{\mu}_{ij} f'(x^{\mu}_{ij})$$

where $t^{\mu}_{ij} = \left[f'(x^{\mu}_{ij})\right] z^{\mu}_{ij}$ and z^{μ} is the fixed point of the error back-propagation GCNN with dynamical equation

$$\dot{z}^{\mu}_{ij} = -z^{\mu}_{ij} + \Sigma_{kl} A(k,l;i,j) z^{\mu}_{kl} + E^{\mu}_{ij} \qquad (2)$$

This net has the same topology as the original one, with transposed weight tensor, a linear output function, and errors E_{ij} as biases.

System (2) has the same fixed points as (1), with the same eigenvalues [HER91]. However, convergence of the back-propagation network is not guaranteed by stability of fixed points of (1), because the output function is linear. For this reason, I added a piecewise linear output function $g(x) = 0.5(|x+K|-|x-K|)$; in this way we obtain the same back-propagation system while in the linear region, i.e. in a neighborhood of the solution (that can be made large for large enough K), but the network, whose dynamical equations are therefore written as

$$\dot{z}^{\mu}_{ij} = -z^{\mu}_{ij} + \Sigma_{kl} A(k,l;i,j) g(z^{\mu}_{kl}) + E^{\mu}_{ij} \qquad (2')$$

is now stable whenever the original one is.

When learning is accomplished on a sequential computer, RBP is rather slow, because it needs thousands of steps involving each the relaxation of two networks. However, this algorithm was chosen because of the success of ordinary back propagation in feed-forward nets. Thinking of real-life realizations, RBP may be implemented in hardware so that one iteration step of the algorithm lasts only a few time constants of the electronic circuit, so that learning time is actually governed only by the frequency of presentation of patterns.

Simulation results - Boolean and continuous function approximation was tried on 1-neighborhood planar networks (type 1, figure 1) and on layered systems of one-dimensional networks (i.e. planar networks with selected connections - type 2, fig. 2).

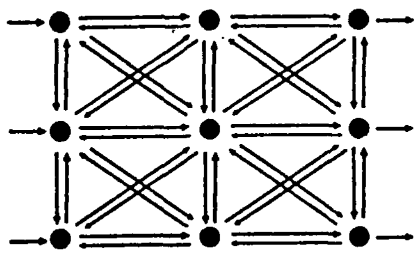
Learning was started with random weights satisfying theorem 4 so as to ensure stability, which was generally preserved during learning, provided that the learning rate was not too large, even if weights eventually violated the condition.

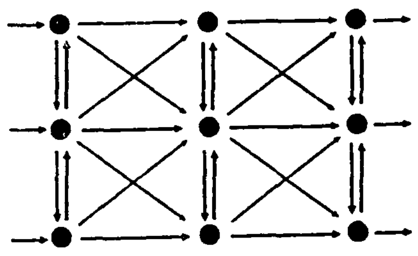
Type 1 nets were soon discarded, because they tend to oscillate very easily and therefore have very long settling times (hundreds of time-constants).

Type 2 networks, instead, proved capable of approximating boolean and continuous functions. A 2×2 network was taught to compute logical AND and XOR of its inputs (fig. 3); 3×3 nets with both topologies of fig. 2(a) and (b) were able to approximate sections of sinusoids.

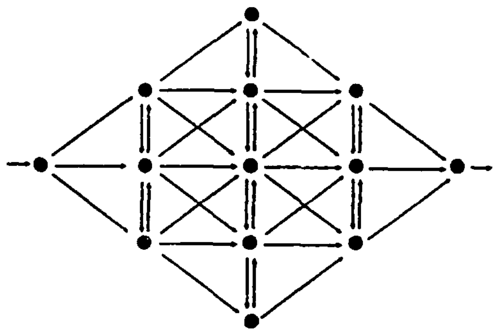
Conclusions and perspectives - The results reported in this work are enough to say that new applications may open up for Cellular Neural Networks, in the fields of approximation, reconstruction of signals, classification. More extensive simulation is in progress in order to inquire into the performance of such networks when confronted with traditional fully connected networks.

figure 1



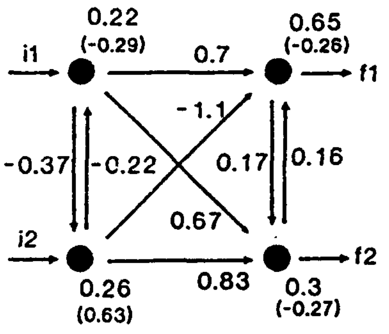


(a)



(b)

figure 2



i1	i2	f1 (XOR)	f2 (AND)
−1	−1	−0.99	−0.97
−1	1	0.99	−0.50
1	−1	0.99	−0.50
1	1	−0.96	0.94

figure 3 XOR/AND network: weights are written near connections and cells; biases I are in brackets.

References

[BAL92] M. Balsi, to appear in Int. J. Circ. Th. Appl.
[BAR92] A. Barone, M. Balsi, V. Cimagalli, in CNNA-92
[CNNA90] *Proceedings of IEEE International Workshop on Cellular Neural Networks and their Applications*, Budapest, Hungary, Dec.16-19, 1990
[CHU88] L.O. Chua, L. Yang, IEEE CAS-35(10), 1257 (1988)
[CHU90] L.O. Chua, T. Roska, IEEE CAS-37(12), 1520 (1990)
[HAR91] H. Harrer, J.A. Nossek, F. Zou, Technische Universität München, rep. TUM-LNS-TR-91-1
[HER91] J. Hertz, A. Krogh, R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991
[HIR89] M.W. Hirsch, Neural Networks 2, 331 (1989)
[PIN87] F.J. Pineda, Phys. Rev. Lett. 59(19), 2229 (1987)
[ROS90] T. Roska: L.O. Chua, in [CNNA90]
[ROS92] T. Roska, Hungarian Academy of Sciences, rep. DNS-1-1992
[SZO91] P. Szolgay, T. Kozek, Hungarian Academy of Sciences, rep. DNS-10-1991
[TAN90] S. Tan, J. Hao, J. Vandewalle, in [CNNA90]
[ZOU91] F. Zou, J.A. Nossek, IEEE CAS-38(6), 675 (1991)

INHOMOGENEOUS CELLULAR NEURAL NETWORKS: POSSIBILITY OF FUNCTIONAL DEVICE DESIGN.

Yuri ANDREYEV, Yuri BELSKY,
Alexander DMITRIEV, Dmitriy KUMINOV
Institute of Radio Engineering and Electronics
of the Academy of Sciences
Marx Avenue 18, 103907 Moscow, Russia
Tel.*(095)-203-48-17; Fax*(095)-203-84-14; Email dmitr@ire.msk.su

Abstract.

A possibility of designing functional devices for various applications in terms of unified technology on the basis of CNNs is discussed on examples of an associative memory device, complex oscillations generator, devices for chaotic memory scanning and pattern recognition.

1. INTRODUCTION.

One of the advantages of analog (CNN)cellular neural networks [1] is a possibility of their realization as chips.

But CNNs with relatively small number of elements may also be promising. Even if all the elements are coupled with each other, the total number of connections is not excessive here.

Then an interesting possibility to design chips of integral functional devices for various applications in terms of unified technology on the basis of CNNs with inhomogeneous couplings, appears.

For example, a possibility to use neural networks as A/D converter and circuits for solving linear programming problems were discussed in [2].

This approach can also be used for designing various classifiers [3].

We will discuss in this report an application of this approach to the design of devices for chaotic memory scanning and pattern recognition.

2. ASSOCIATIVE MEMORY, CHAOTIC SCANNING AND PACEMAKER.

Two devices are necessary to organize chaotic memory scanning: a memory [4] itself and some external generator or pacemaker to push the system from one memorized pattern to another.

2.1. Associative memory on the basis of an inhomogeneous CNN.

To store images into a neural network, we have to find an algorithm of forming the matrix of couplings or templates, providing correspondence between equilibrium states of a CNN circuit and memorized patterns.

ADALINE algorithm [5] was chosen as a learning rule (an algorithm of coupling matrix forming). This algorithm was chosen for the next reasons: it is simple in realization and its matrix of couplings is not symmetric.

In general, the matrix of couplings is not local.But demands on the locality of the matrix of couplings can be lowered for the case of a little number of cells and images. This case of a little number of cells and patterns is important not only as an example, but for classifier design also.

2.2. Generator of complex oscillations with amplitude modulation.

The dynamics of a non-autonomous harmonically driven CNN composed of two cells was studied in details in [6]. A chaotic attractor was found in this non-autonomous system.

Such complex dynamics can obviously be obtained in an autonomous system, replacing an external harmonic signal by the signal from a generator composed of two cells. The dynamics of the four cell-CNN is described by:

$$\begin{aligned}
&dY/dt + \mathbf{Y} = \mathbf{T} * \mathbf{F}(\mathbf{Y}),\\
&\mathbf{Y} = \{\ y(1),\ y(2),\ y(3),\ y(4)\ \}\\
&\mathbf{F}(\mathbf{Y}) = \{\ f(y(1)),\ f(y2)),\ f(y(3)),\ f(y(3))\ \}, \qquad (1)\\
&f(x) = (|x + 1| - |x - 1|) / 2
\end{aligned}$$

To design a pacemaker that could be used to control memory scanning a neural generator in which a large amplitude chaotic oscillations are interchanged by oscillations with a small amplitude near zero value, should be constructed.

We modify then the system (1) to a neural network of Fig.1. Two additional controlling cells 5 and 6 are introduced here. The dynamics of the cells 5 and 6 are described by equations

$$\alpha(t)\cdot dx/dt + x = f(u),$$
$$\alpha(t) = \alpha_0 + \alpha_1 \cdot \sin(2\pi t/T) \qquad (2)$$

where u is an output signal from the cells 1 or 3 for the cells 5 or 6 respectively, T is a period of controlling signal for parameter $\alpha(t)$.

With an increase of the parameter $\alpha(t)$ chaotic oscillations disappear in the system.

We can obtain the needed control signal (Fig.2) for the memory device with the help of high-pass filter composed of neural cells.

Applying the signal from the pacemaker to the associative memory device we obtain the device for chaotic scanning of memory. Fig.3 shows the regime of memory scanning for CNN composed of nine cells with tree stored patterns V1, V2 and V3.

It should be noted that chaotic memory scanning may also be organized in some other way [7].

3. PATTERN RECOGNITION.

By pattern recognition we mean the following.Let an external signal is applied to a neural system, in which chaotic scanning of memory is organized. If the external signal corresponds to one of the patterns, stored in the neural network (memory device),the system must stabilize itself in the state, associated with that pattern. Otherwise, it should continue chaotic memory scanning.

For the organization of pattern recognition we introduce a local feedback between the memory and pacemaker. This feedback turns on and destroys the complex oscillations of the pacemaker only if a given equilibrium state of the memory device corresponds to the pattern being recognized.

The feedback can be described by the relation

$$\alpha(|\mathbf{Y} - \mathbf{Z}|) = k/(|\mathbf{Y} - \mathbf{Z}| + \varepsilon), \qquad (3)$$

where k and ε are some fixed parameters and $\varepsilon \ll 1$; vector **Z** stands for an external signal or external pattern; vector **Y** describes an internal state of associative memory; $|\,*\,|$ is a vector norm. $\alpha(|\mathbf{Y} - \mathbf{Z}|)$ is added to $\alpha(t)$ (section 2.2).

We organize pattern recognition only on one component of Z.

$$\alpha(|\mathbf{Y} - \mathbf{Z}|) = k/(|y(2) - z(2)| + \varepsilon), \qquad (4)$$

where y(2) is the second component of **Y**, z(2) is the second component of **Z**.

Fig.4 shows the trajectory of the state variable y(2) for a pattern recognition regime.

4. CONCLUSION.

So we discussed the possibility of applying CNNs and their combinations to the design of various functional devices. This approach was discussed on the examples of associative memory, complex oscillation generator, chaotic memory scanning and pattern recognition. The advantage of this approach is in the use of universal CNN chips for different purposes.

REFERENCES

1. L.O.Chua, L.Yang, "Cellular Neural Networks", IEEE Transactions on Circuits and Systems, 1988, V.35, pp.1257-1272, pp.1273-1290.
2. D.W.Tank, J.J.Hopfield,"Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit,and a Linear Programming Circuit", IEEE Transactions on Circuits and Systems, 1986, V.33, pp.533-541.
3. D.E.Rumelhart, G.E.Hinton, R.G.Wiliams, "Learning Representation by Back-Propagating Errors", Nature, 1986, V.323, pp.533-536.
4. J.J.Hopfield, " Neural Networks and Physical Systems with Emergent Collective Computational Abilities ", Proceedings National Academy of Sciences USA, 1982, V.79, pp.2554-2558.
5. J.S.Dencer ," Neural Networks: Modeling and Adaptation ", Physica D, 1986, V.22, pp.216-232.
6. F.Zou, J.A.Nossek,"A Chaotic Attractor with CNNs", IEEE Transactions on Circuits and Systems, 1991, V.38, pp.811-812.
7. A.Dmitriev, L.Jdanova, D.Kuminov, "The Simplest Neural-Like Systems with Chaos",Proceedings of the International Conference on Noise in Physical Systems and 1/f Fluctuations , November , 1991, Kyoto, Japan, pp.501-504.

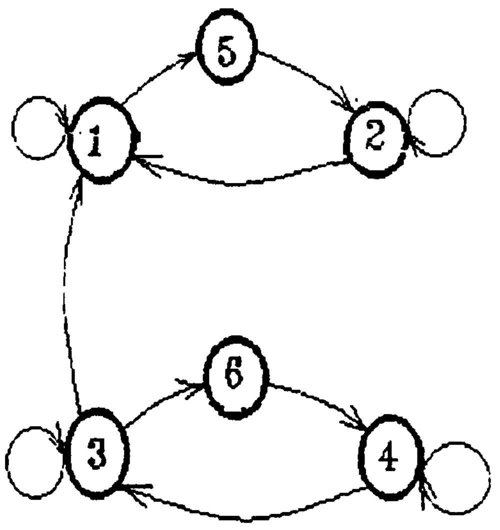


Fig.1 The four-cell CNN with two control cells 5 and 6. Couplings between cells are $T_{11} = T_{22} = 2.0$, $T_{33} = T_{44} = 1.2$, $T_{15} = T_{36} = 1$, $T_{52} = -T_{21} = 1.2$, $T_{64} = -T_{43} = 2.1$, $T_{31} = 3.5$.

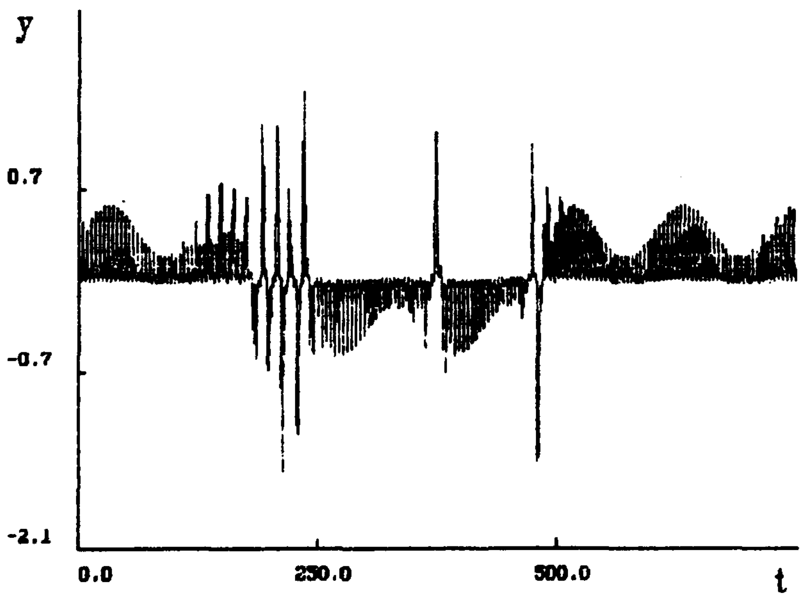


Fig.2. Control signal for the "memory" device.

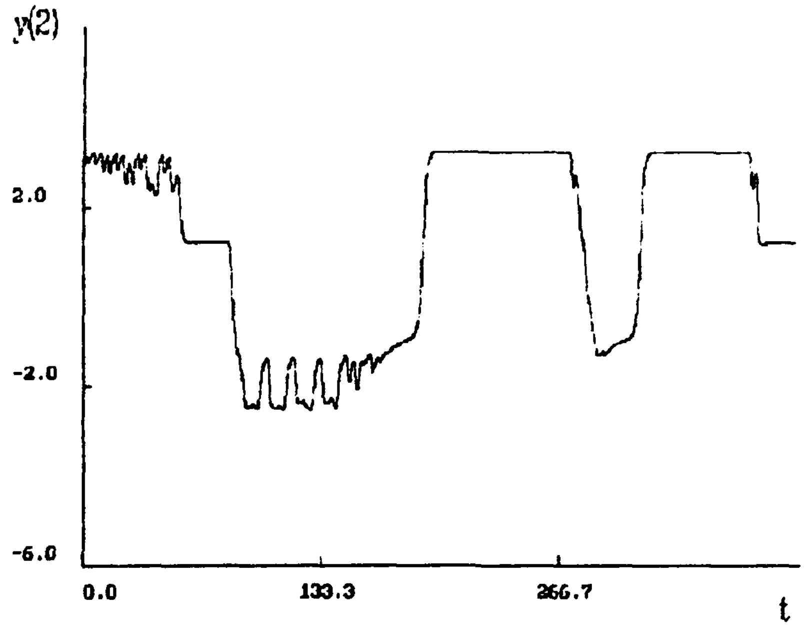


Fig.3. Chaotic memory scanning. Memory device falls onto memorized pattern V1 for $0 < t < 45$, $200 < t < 270$, $310 < t < 360$; onto V2 for $90 < t < 190$, $280 < t < 300$; onto V3 for $45 < t < 90$, $360 < t < 400$.

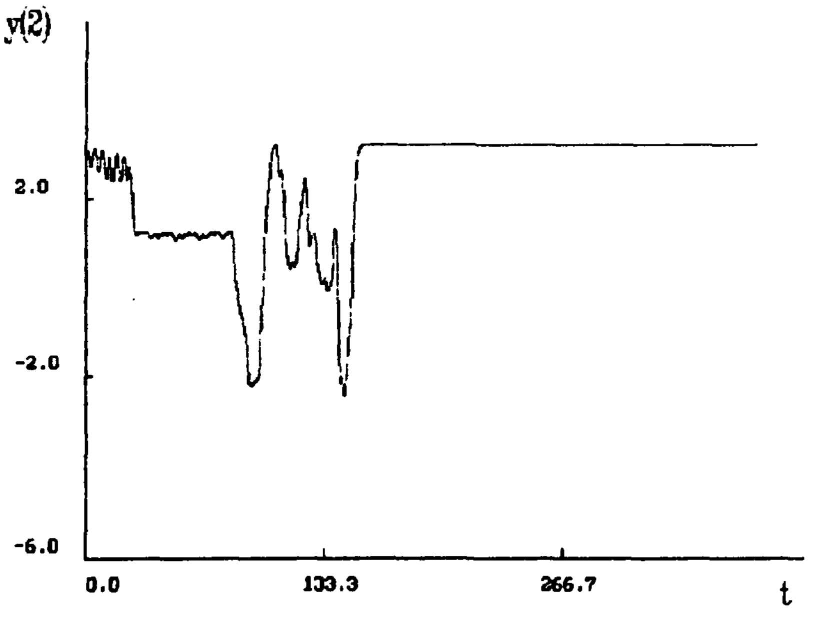


Fig.4. Pattern recognition: $k = 0.02$, $e = 0.002$. Z is an arbitrary external pattern for $0 < t < 140$ and $Z = V1$ for $t > 140$.

Analog VLSI Implementation of Cellular Neural Networks

J.L. Huertas, A. Rodríguez-Vázquez and S. Espejo

Department of Analog Design
Spanish Microelectronics Center (Centro Nacional de Microelectrónica)-Universidad de Sevilla
Edificio CICA, C/Tarfia sn, 41012-Sevilla, SPAIN
Tel. 34 5 4623811; Fax 34 5 4624506
email: angel@cnm.us.es

Abstract

This paper covers the design of continuous-time (CT) and discrete-time (DT) Cellular Neural Networks (CNN) using analog VLSI circuit techniques. A new cell model is proposed which exhibits advantages for reduced area and power consumption CNN implementations. This model is very well suited for implementation in current domain, which is also important to avoid the need for current-to-voltage dedicated interfaces in image processing tasks with photosensor devices. Cell design relies on exploitation of current mirrors for the efficient implementation of both linear and nonlinear analog operators. These cells are simpler and easier to design than those found in previously reported CT and DT-CNN devices. Basic design issues are covered, together with discussions on the influence of non-idealities and advanced circuit design issues as well as design for manufacturability considerations associated with statistical analysis.

Introduction

Cellular Neural Networks (CNN) consist of arrays of elementary processing units (*cells*), each one connected only to a set of adjacent cells (*neighbors*). This local connection property makes CNN physical design easy (specially for the class of *translationally invariant* CNNs, where all inner cells are identical), allowing increased cell density per silicon area. Also, programmability issues can be easily incorporated to *translationally invariant* CNNs without significant extra routing cost, by just adding several global control lines, one per weight.

CNN properties, and its application to image processing, pattern recognition, motion detection, etc., have been covered in different papers, for instance [Chua88a, 88b, 90, 91a, Noss92, Mats90a, 90b, 90c, Mats91]. This paper focuses on CNN VLSI implementation, of which little literature is available [Cruz91, Halo92, Harr92]. For implementation purposes, analog CNNs can be classified into *continuous-time* (CT) [Chua88b] and *discrete-time* (DT) [Rodr90, Harr92] models. Each model type is described by a set of nonlinear dynamic equations, one per cell, whose associated equilibrium state distribution determines the network computational properties. Previously reported CT-CNN IC design approaches focused on the use of g_m-C techniques for the implementation of the Chua-Yang's CNN cell circuit model [Chua88a]. Proposed discrete-time realizations focused on a very similar cell circuit model, for which MOSFET-C techniques and fully differential high-output impedance opamps have been considered [Harr92]. In all cases, input signals are voltages, while internal signals can either be voltages or currents. Since primary output of image sensor devices (*phototransistors* [Sayl91]) is current, the need arises to convert these outputs to voltage, thus complicating CNN interface design for image processing tasks. Also, electrical cell design is

0-7803-0875-1/92 3$.00 ©1992IEEE

not easy because different dynamic ranges for the internal voltages and currents must be considered to guarantee a reduced influence of the MOS transistor nonlinearities. Finally, operation speed is not optimum because the combination of internal voltage and current signals results in internal high-impedance nodes, and hence, large time constants.

Here, a new CNN cell model is presented and implementation techniques are discussed for both CT and DT CNNs using current-mode techniques. Cell complexity is shown to be much smaller than for previous approaches. Also, design is very simple (only two sizing equation are needed) and the speed/power figures are very good due to the lack of internal high-impedance nodes.

Chua-Yang's CNN Model:Analog Implementation.

Fig.1 shows the Chua-Yang's CNN cell circuit model, whose dynamic is given by:

$$\tau\frac{dx^c}{dt} = -x^c(t) + D^c + \sum_{d \in N_r(c)} \{A^c_d y^d(t) + B^c_d u^d\} \qquad \forall c \in \mathcal{GD} \tag{1}$$

where $N_r(c)$ represents the *cell neighborhood*, including cell c itself, $\mathcal{GD}$ is the net *grid domain*, and the *cell outputs* (y^d) are obtained from the *state variables*, x^d, by the following nonlinear function,

$$y^d(t) = \frac{1}{2}(|x^d(t) + 1| - |x^d(t) - 1|) \tag{2}$$

The input (B_d^c) and output (A_d^c) weights are called *control* and *feedback* parameters, respectively, and D^c is the *offset* parameter. Computational properties of Chua-Yang's model rely on its ability to yield, for $A^c_c > 1$, two stable equilibrium points separated by an instability region, and in the possibility of modifying the stable point attraction regions by changing the neighbor contributions,

$$I = D^c + \sum_{\substack{d \in N_r(c) \\ d \neq c}} \{A^c_d y^d(t) + B^c_d u^d\} + B^c_c u^c \tag{3}$$

This is illustrated in Fig.2, showing the $\tau(dx^c/dt)$ vs x^c characteristic for the different qualitative situations possible. The displayed dynamic routes show that the attraction region for the equilibrium point on the right becomes narrower for I decreasing; for $I=I_3$, this point becomes virtual. A similar situation happens for the equilibrium point on the left, in case I increases.

Analog VLSI implementation of (1) must handle different variation ranges for the state and the output variables, as illustrated in Fig.2: while the output variables change inside [−1,1], state variable excursions are constrained by the following normalized maximum value [Chua88a],

$$x_{max} = 1 + |D^c| + \sum_{d \in N_r(c)} \{|A^c_d| + |B^c_d|\} \tag{4}$$

which, for typical templates, ranges between 5 and 10. Thus, in the previously reported g_m-C circuits [Cruz91, Halo92], where all resistive components in Fig.1 are realized by differential input transconductors, largely different biasing conditions and design equations must be considered for the transconductors, thus complicating the sizing process and yielding non-optimum power consumption and area figures.

Extended Range CNN Models

A new CNN model which preserves qualitative properties of the Chua-Yang's model and yields identical variation ranges for both state and output variables is proposed as follows,

$$\tau\frac{dx^c}{dt} = g[x(t)] + D^c + \sum_{d \in N_r(c)} \{A^c_d y^d(t) + B^c_d u^d\} \qquad \forall c \in \mathcal{GD} \tag{5}$$

where $g(\bullet)$ is a three pieces piecewise linear characteristics given by,

$$g(x^c) = \lim_{m \to \infty} \begin{cases} -m(x^c+1)+1 & x^c < -1 \\ -x^c & otherwise \\ -m(x^c-1)-1 & x^c > 1 \end{cases} \tag{6}$$

Convergence towards binary states in the new model is illustrated in Fig.3. It is seen that dynamic routes converge to binary states for all possible qualitatively different cases.
A similar full range model is found for discrete time CNNs,

$$y^c(n+1) = f\left[D^c + \sum_{d \in N_r(c)} \{A^c_d y^d(n) + B^c_d u^d\}\right] \qquad \forall c \in \mathcal{GD} \tag{7}$$

which involves only input and output variables. In this model, convergence towards binary output requires that $f(\bullet)$ be sigmoidal and that the equivalent slope at the origin, $A_c^c f'(0)$, be larger than 1 to ensure *regenerative* behavior. This can be achieved by using either a *soft nonlinearity* such as that in (2) and $A_c^c > 1$, or by using a *comparator*,

$$f(z) = \begin{cases} 1 & for \quad z > 0 \\ -1 & for \quad z < 0 \end{cases} \tag{8}$$

and $A_c^c = 1$, as proposed in [Harr92]. Convergence towards binary outputs for the DT-CNN model is illustrated in Fig.4.

Current-Mode CNNs Conceptual cells

Signal *summation, scaled replication, integration, delay* and *nonlinear transformation* are the analog operators required to implement CNNs. Summations are performed in current-mode by routing currents to a common node. Remaining operators are realized using a very simple analog building block: *the current mirror.* A current mirror yields linear current scaling via a simple, yet ingenious, concept involving nonlinearity cancellation between *matched* transconductors. Fig.5 shows current-mirror based realizations for the different CNN model cells given above using a generic, abstract three terminal transconductor (Fig.6) such that,

$$\begin{aligned} i_2 &= Pu(v_1, v_2) \\ i_1 &\ll i_2 \end{aligned} \tag{9}$$

where $u(\bullet)$ is assumed invertible, at least in v_1, the characteristic is parameterized by a designer-controlled scale factor P, and the device is assumed to enter a *cut-off* region, with i_2=0, when input voltage is below a *cut-in* value. Fig.5(a) applies for all models discussed in the paper. Switches labelled S_t and $\overline{S}_t$ are used for cell initialization purpose. Fig.5(b) is a schematic for the cell's dynamic part in the Chua-Yang model. Corresponding schematics for the full range CT model is shown in Fig.5(c). Switch R_c in both schematics is used for initialization purposes. It is seen that the full range model gives simpler circuits than the Chua-Yang model. Fig.5(d) shows the schematics of the cell's dynamic part for the DT CNN model.

In the simplest case, the generic transconductor contains only one transistor. Besides, only one MOS transistor is required to implement an analog switch with zero ON offset and very low OFF leakage (about 10pA). This infers that for this simplest case, DT CNN cells of Fig.5 are implemented in CMOS using only 18 transistors: this is an important advantage when compared to previous approaches for DT CNNs [Harr92], where 106 transistors are required for a similar cell. For CT Chua-Yang CNNs, complexity (measured in number of transistors) of the current-mode circuit is similar to previous g_m-C implementations [Cruz91]. However, this complexity decreases in current mode implementations of the full range model (down to 18 transistors per cell). Despite the actual model considered, the design equations for current mode CNN ICs are much simpler than for previously reported techniques because of the intrinsic current mirror nonlinearity cancellation. Also, current mode CNNs are faster and allow larger pixel densities.

CMOS Current Mode CNN Design Issues

Current mode CNN operation is degraded by both random and systematic sources of error. Random errors are due to statistical variations of the technological parameters across the die and can be attenuated using large devices, careful layout, and proper bias generation and distribution. Systematic errors are, on the other hand, corrected by proper transconductor choice and transistor sizing. Two major systematic error sources can be identified: **a)** Input-output voltage mismatching at the bias point, defined as the point where transconductors sink only their bias currents; **b)** Finite R_o/R_{in} ratios, where R_o represents the transconductor output resistance and R_{in} holds for the input resistance (for an ideal mirror, R_o/R_{in} should be infinitely large).

Static Nonidealities and Sizing Equations: Voltage mismatching produces current offset, originated by the transconductor current dependence on the output terminal voltage. For the MOS transcoductors in the paper, this offset is eliminated by forcing the same current density in the mirror input transconductor and loading devices. Finite R_o/R_{in} ratio causes current gain error due to spurious current division at the mirror input and output nodes. The error is especially significant if small dimension single-transistor transconductors are used, due to the very low Early voltages associated to short channel transistors. It can be corrected by increasing device size (channel length) but this does not yield optimum area and speed for CNN implementations in scaled down technologies. For improved R_o/R_{in} figures with short channel devices, either *cascoded* transconductors, or feedback transconductors, or a combination of both must be used [Greg86, Sack90]. Figs.7(a) and (b) show the two CMOS mirror structures considered for CNN design, including the complementary devices used for biasing (dashed lines). We have chosen n-channel devices for the transconductors since their K values are larger than those of the p-channel $(K_n>K_p)$, yielding more area-efficient CNNs. *Design parameters* are displayed in the figures: W, L_n, L_p and V_{CAS}. Fig.7(c) shows a circuit to provide this cascode voltage. We assumed that Early voltages are proportional to the channel length: $V_{An}=\alpha_n L_n$, $V_{Ap}=\alpha_p L_p$. In Fig.7(a) channel lengths are different for NMOS (L_n) and PMOS (L_p), to obtain equal nominal Early voltages for both devices, and, hence, optimize R_o/R_{in}. In the case of Fig.7(b), this figure is intrinsically much larger and all channel lengths are made equal ($L_n = L_p$) for simpler design. To achieve simplicity, we have also assumed that all transistors have the same channel width, W. Table 1 gives sizing equations for Figs.7(a) and (b). These equations are intended to ensure that the mirrors handle the whole input current range with minimum distortion, and using the smallest possible devices. The W expressions given in the table correspond to a bias current I_Q; W values for larger currents (associated either to output transconductors where gain is larger than unity or to input transconductors with bias current SI_Q) are calculated taking into account the requirement for equal current density in all transconductors.

Table 1: Sizing Equations for CMOS Mirrors

	Channel widths ($W = W_n = W_p$)	Lengths	Bias voltage
single device mirror	$W = \frac{4I_Q L_n}{(V_{DD} - V_{SS} - V_{Tn})^2}\left[\sqrt{\frac{1}{K_n}} + \sqrt{\frac{\alpha_n}{2K_p\alpha_p}}\right]^2$	$L_p = \frac{\alpha_n}{\alpha_p} L_n$	
cascode mirror	$W = \frac{16 I_Q}{K_n V_{Tn}^2} L_n$	$L_p = L_n$	$V_{CAS} \approx V_{SS} + 2V_{Tn}$

Note that the sizing equations are parameterized by L_n, which is chosen by the designer to control R_o/R_{in} and the channel area. Fig.8 shows the current gain error per cell versus the total cell area for a full range CT CCD CNN using single and cascode mirrors in n-well 1.6μm CMOS: the top family is for the single transistor mirror, and the bottom family for the cascode. Parameter is the rail current I_Q, which varies from 0.25μA to 128μA. As it can be seen, simple current mirror requires large area to achieve acceptable error figure. On the other hand, cascode mirrors allow using short channel-length devices, and, thus result in much higher area efficiency.

Dynamic Nonidealities: It can be seen that the time constant of the CT models depends on the state variable value; for instance, for the Chua-Yang model the following is obtained,

$$\tau = \frac{C}{\sqrt{2\beta\,(x^c + SI_Q)}} \tag{10}$$

Thus, transient is faster if the cell state is near the black pixel. However, the equilibrium points are exactly the same as for the nominal case, corresponding to the solution of the equation,

$$h^c = -x^c + \frac{A_c^c}{2}\left(|x^c(t) + I_Q| - |x^c(t) - I_Q|\right) + I = 0 \tag{11}$$

Also, the sign of the state variable derivative (equivalently, the sign of h^c) for a given I has exactly the same dependence on x^c as for the nominal case, and, as a consequence, actual dynamic routes are identical to the nominal.

Another dynamic error arises in the analog switches due to the necessity to evacuate the MOS channel charge during the switch turn-off transient process. This error, generically called *feedthrough error* [Eich89], is very large (up to 20% and more) if small geometries transconductors are used. A simple technique to attenuate this error is the inclusion of an additional capacitor at the switch output node. Since neither linearity nor accuracy in the capacitance is required for this purpose, a shorted transistor can be used. This is feasible for standard digital CMOS technologies (having only one poly layer) and require less area than typical capacitors in two poly technologies (Poly1-SiO_2-Poly2), since channel oxide is usually thinner than interpoly oxide. The use of this technique can easily lower the error about one order of magnitude. Also, since only two of these devices are required in each CNN cell, area penalization is not severe. Much lower feedthrough error is achieved using a dummy transistor, and automatically tuning the delay between the switching device clock signal and the dummy device clock signal [Espe92]. In this manner errors as little as 0.3% have been measured on silicon prototypes. Also, the extra monitoring and control circuitry required for this automatic tuning can be shared for all cells in the network, so that area penalty is not severe at the network level.

Bias Current Selection: A crucial issue that has not been discussed yet is the election of the bias current I_Q. The geometry factor (W/L) of the transistors in Figs.7(a) and (b) increases monotonically with I_Q. Furthermore, static gain error due to finite R_o/R_{in} values also increases with I_Q. In addition, power dissipation is proportional to the bias current. For these reasons, a bias current as small as possible should be choosen. The issue is to identify the minimum feasible rail current value. Lowest limit of the bias current is certainly established by leakage (about 10pA in standard CMOS). However, a more restrictive bound exists due to MOS transistor mismatch [Pelg89] and Early voltage (V_A) degradation with channel length. These phenomena, which actually limit the minimum devices geometries, also restrict the minimum current.

Mismatch is mainly produced by variations of the threshold voltages (V_T) and large signal transconductance ($\beta=\mu C_{ox}W/L$) of equally designed transistors in the same chip. Standard deviation $\sigma(V_T)$ and ratio $\sigma(\beta)/\beta$ can be expressed in terms of two statistically different contributions: a white espectra component, that accounts for the effect of phenomena with extremely short correlation distance (much less than minimum transistor dimensions), and a second component that reflects the effect of large correlation distance phenomena. The first component is, in a good aproximation, inversely proportional with the square root of the channel area, while the second is proportional to the distance between two equally layed-out devices. Results in [Pelg89] demonstrate that the distance dependent component is negligible for devices with a channel area less than about $100\mu m^2$. For bias currents below about 50µA and cascode mirrors, design equations in Table 1 give devices with channel area well below this bound. Hence, the distance dependent component need not to be considered for current mode CNNs, but only the $\sigma(V_T)$ and $\sigma(\beta)/\beta$ dependence with channel area.

In adition, for a given $\sigma(V_T)$ and $\sigma(\beta)/\beta$, the ratio $\sigma(I)/I$ in MOS transistors is inversely proportional to the gate-source voltage v_{gs}. This means that, once W/L factors have been set to achieve acceptable mismatch levels, current bias can not be decreased too far below the upper bound given by equations in Table 1, since this would produce a low v_{gs} voltage in the bias point, with the corresponding high $\sigma(I)/I$. Hence, mismatch considerations establish bounds for both minimum area and power trends. For example, we have obtained 100% success (out of 30 trials) for a Montecarlo simulation of a connected component detector (CCD) current mode full range CT CNN with 16 cells in a row using unitary transistor geometries of W/L=4µm/3.2µm for both n and p-channel devices, and a bias current I_Q=2µA. Further geometry reduction has not been considered, since minimum contact size (4µm with surrounding diffusion in the 1.6µm n-well technology used) does not allow a significant area reduction anyway. Similar simulations performed on a Chua-Yang model counterpart resulted in lower yield figures, which can be understood by observing that the normalized

value dispersion of the equilibrium points (normalization factor is the distance between nominal equililibrium state positions) is much smaller for the full range model, than for the Chua-Yang model. Hence, larger geometries should be used for increased yield with this model.
Current references for every cell in the network and for every current source within each cell can be generated from common bias voltages. Reference dispersion due to mismatch among the transistors of different current sources did not produce critical results in Montecarlo simulations. Cascode voltage for current mirrors can also be globally distributed. If high noise levels are expected at bias voltages, as may be the case in discrete time implementations, a bias voltage independent current reference [Greg86], shown in Fig.9, can be included in each cell, although a strong area penalization may result. Current reference dispersion will be produced only by local mobility and resistivity variations. Montecarlo simulation of entire CCD systems where cell current references have a standard deviation larger than 5% have shown 100% success (out of 15 trials).

Discussion of Results

Table 2 gives the transistor count and total cell area for different templates, and for both CT models. Cascode mirrors with W=4μm and L=3.2μm are used. For DT implementations area is slightly larger than those for the full range CT model, due to the switches. Although data in Table 2 do not include the area occupied by the initialization circuitry, pixel-densities ranging from 60 to more than 160 cells/mm^2 can be easily achieved (depending on the particular template) if the full range model is used.

Table 2: Cell area and transistor count for different templates and CNN models.

	Full Dynamic Range Area (μm^2)	**Chua-Yang Model** Area (μm^2)	**Full Dynamic Range** Ttor. Count	**Chua-Yang Model** Ttor. Count
C. C. Detec.	5916	12691	40	56
Shadow Detec.	7736	16533	52	68
Borders Extrac.	15471	26291	112	128
Corners Extrac.	16381	28212	120	136
Hole Filling	10921	24774	76	92
Noise Filtering	5460	14258	40	56

Several 1.6μm CMOS prototypes have been designed to probe ideas in the paper: one is DT and can be reconfigured via local logic for different templates, other two prototypes are CT and fixed template (CCD and other for noise removal). Obtained results demonstrate the possibility to achieve very large cell densities with good yield figures by using:

*Full range CNN models, allowing all cell variables to have the same variation ranges and, hence, better area and power consumption figures.

*Intrinsic functional nonlinearity cancellation, giving technology independent circuits and architectures, and allowing the simplification of the IC design process.

*Cascode transistors, to reduce static error terms without area and speed penalization.

*Careful consideration of electrical issues related to net architecture and input/ouput interfacing.

For CMOS technology, current as low as 2μA can be used with reasonable yield, if strong inversion is used. Speed for these low current levels is optimum (about 1μs for 16 cells CCD) due to the small device dimensions and the low impedance of internal nodes. Electrical tunability can be easily incorporated using electrically parameterized transconductors, for instance, differential amplifiers. Digital tunability is direct by switching lines rooted to common nodes. Delay templates are in this way easily incorporated.

References

[Chua88a]L.O. Cua and L. Yang: "Cellular Neural Networks: Theory". *IEEE Trans. Circuits and Systems*, Vol. CAS-35, pp 1257-1272, 1988.

[Chua88b]L.O. Cua and L. Yang: "Cellular Neural Networks: Applications". *IEEE Trans. Circuits and Systems*, Vol. CAS-35, pp 1273-1290, 1988.

[Chua90] L.O. Chua and T. Roska: "Stability of a Class of Nonreciprocal Cellular Neural Networks". *IEEE Trans. Circuits and Systems*, Vol. CAS-37, pp 1520-1527, 1990.

[Chua91a]L.O. Chua and P. Thiran: "An Analytical Method for Designing Simple Cellular Neural Networks". *IEEE Trans. Circuits and Systems*, Vol. CAS-38, pp 1332-1341, 1991.

[Chua91b]L.O. Chua and B. Shi: "Multiple Layer Cellular Neural Network: A Tutorial". in *Algorithms and Paralell VSLI Architectures*, A.J. Van der Veen and F. Deprettc, Eds. North Holland, 1991.

[Cruz91] J.M. Cruz and L.O. Chua: "A CNN Chip for Connected Component Detection". *IEEE Trans. Circuits and Systems*, Vol. CAS-38, pp 812-817, 1991.

[Eich89] C. Eichenberger: *"Charge Injection in MOS-Integrated Sample-and-Hold and Switched-Capacitor Circuits"*. Hartung-Gorre Series in Microelectronics, Vol. 3, 1989.

[Espe92] S. Espejo, A. Rodríguez-Vázquez, R. Domínguez-Castro and J.L. Huertas: "An Adaptive Scheme for Feedthrough Cancellation in Switched-Current Techniques". *Proc. IEEE 1992 Midwest Symp. Circuits and Systems*, 1992 (to appear)

[Halo92] K. Halonen et al: "VLSI Implementation of a Reconfigurable Cellular Neural Network Containing Local Logic". *Int. J. Circuit Theory Applications*, 1992 (to appear)

[Harr92] H. Harrer et al.: "An Analog Implementation of Discrete-Time Cellular Neural Networks". *IEEE Trans. Neural Networks*. Vol. 3, pp 466-476, 1992.

[Mats90a]T. Matsumoto et al.: "CNN Cloning Template: Connected Component Detector". *IEEE Trans. Circuits and Systems*, Vol. CAS-37, pp 633-635, 1990.

[Mats90b]T. Matsumoto et al.: "CNN Cloning Template: Hole Filler". *IEEE Trans. Circuits and Systems*, Vol. CAS-37, pp 635-638, 1990.

[Mats90c]T. Matsumoto et al.: CNN Cloning Template: Shadow Detector". *IEEE Trans. Circuits and Systems*, Vol. CAS-37, pp 1070-1073, 1990.

[Mats91] S. Matsui and T. Okumoto: "A Two-Dimensional Segmentation-Free Learning Recognition System by a Cellular Automaton Array using Eigenvectors of the Second Moment Matrix". *IEICE Transactions*, Vol. E-74, pp 2432-2440, 1991.

[Noss92] J.A. Nossek et al.: "Cellular Neural Networks: Theory and Circuit Design". *Int. J. Circuit Theory Applications*, 1992 (to appear)

[Pelg89] M.J.M. Pelgrom et al.: "Matching Properties of MOS Transistors". *IEEE J. Solid-State Circuits*, Vol. SC-24, pp 1433-1440, 1989.

[Rodr90] A. Rodríguez-Vázquez, R. Domínguez-Castro and J.L. Huertas: "Accurate Design of Analog CNN in CMOS Digital Technologies". *Proc. 1990 IEEE Int. Workshop on Cellular Neural Networks and Their Applications*, pp 273-280, 1990.

[Sayl91] A.H. Sayles and J.P. Uyemura: "An Optoelectronic CMOS Memory Circuit for Paralell Detection and Storage of Optical Data". *EEE J. Solid-State Circuits*, Vol. SC-26, pp 1110-1115, 1991.

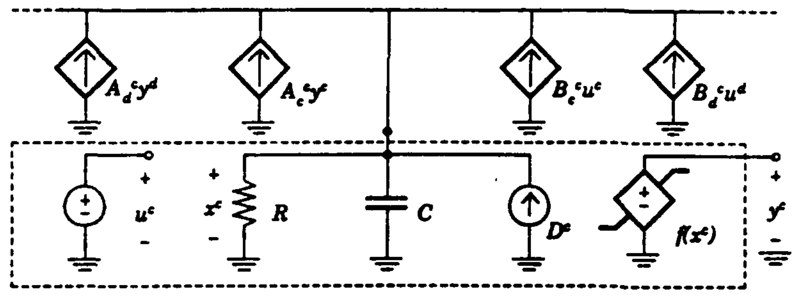


Figure 1

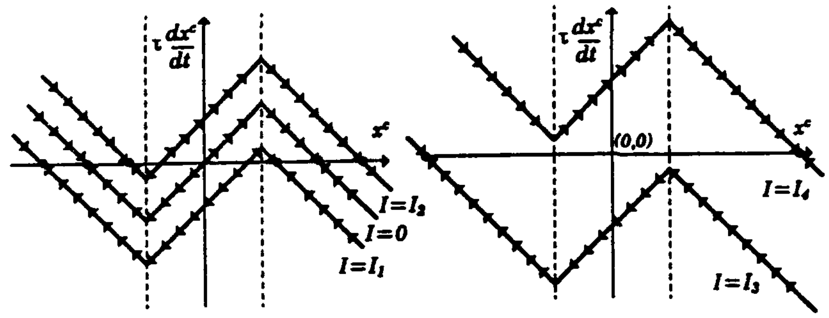


Figure 2

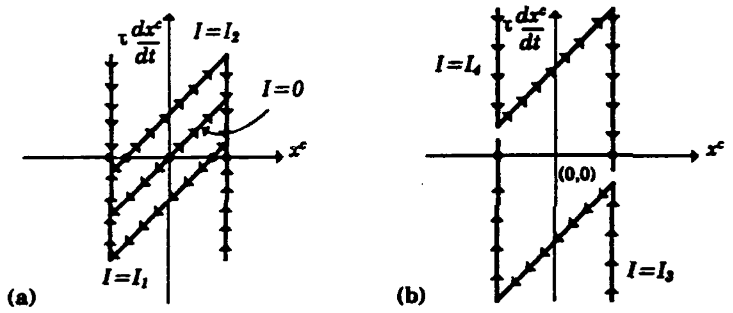


Figure 3

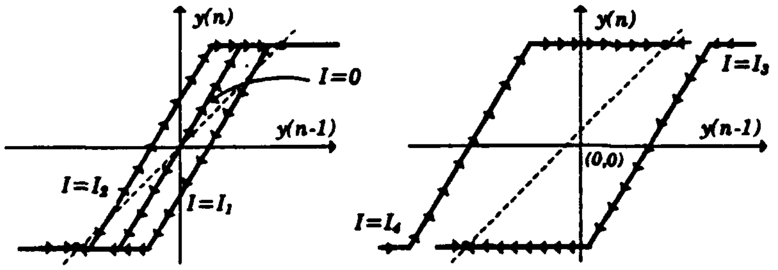


Figure 4

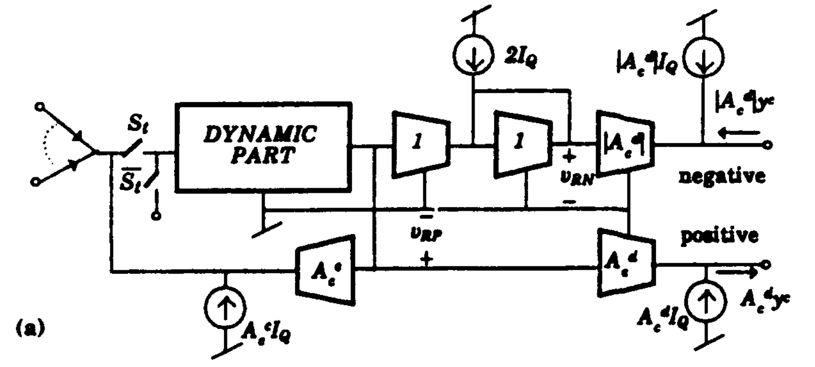
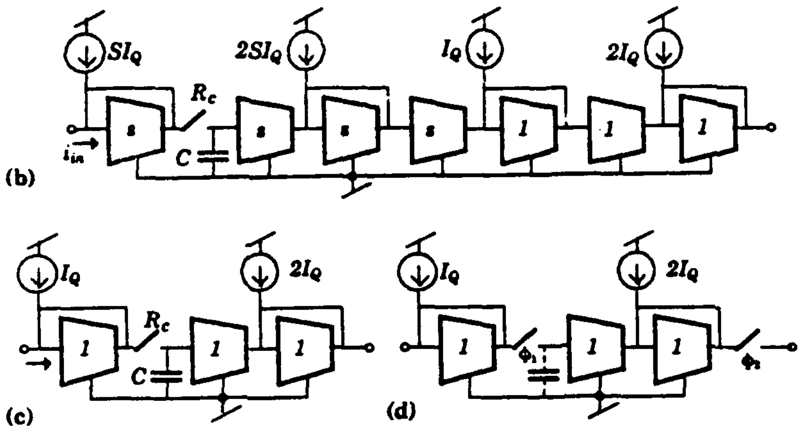


Figure 5



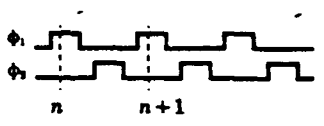


Figure 5

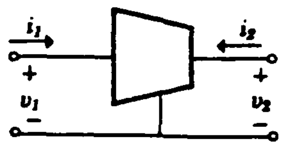


Figure 6

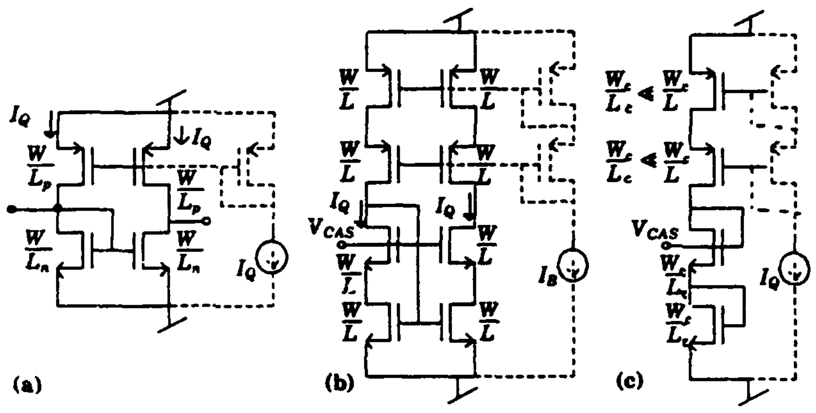


Figure 7

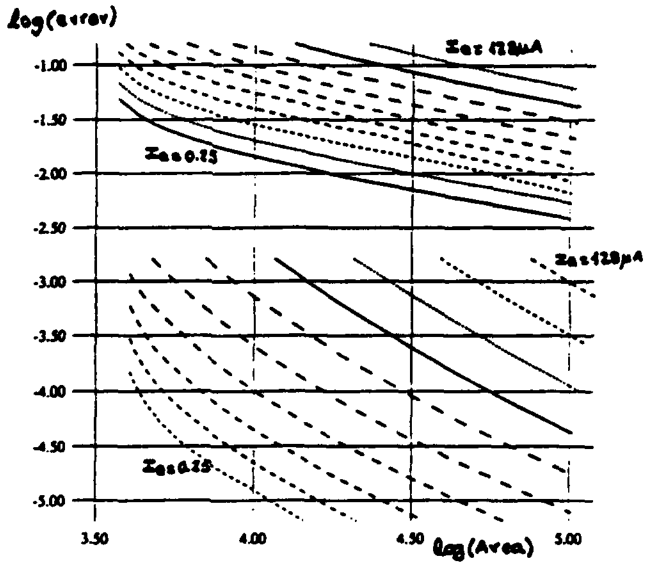


Figure 8

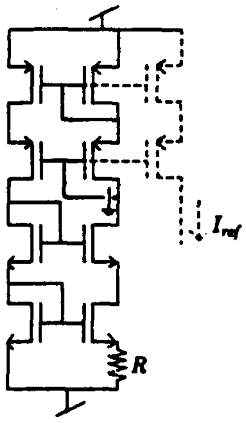


Figure 9

DESIGN OF A CMOS ANALOG PROGRAMMABLE CELLULAR NEURAL NETWORK

G. F. Dalla Betta, S. Graffi, G. Masetti, Zs. M. Kovács
D.E.I.S, University of Bologna
Viale Risorgimento 2 - 40136 Bologna - ITALY
Fax: +(39) 51 644-3073
graffi@deis01.cineca.it, masetti@deis05.cineca.it, kovacs@deis04.cineca.it

Abstract

The design of a CMOS analogically programmable cellular neural network is reported. The electrical characteristics of the basic building blocks are analysed and discussed. Additionally, some performances of a 10x10 CNN are reported.

1 INTRODUCTION

In recent years a new and alternative approach to the classic neural network topologies, called Cellular Neural Networks (CNN's), was introduced by Chua et al [1, 2]. CNNs exhibit several interesting features and properties: each cell is locally connected only to its neighbours; the processed signals are analog in nature; the circuit architecture is space invariant and this renders the CNNs particularly suited for an implementation in a VLSI technology, like CMOS;

Up to now, only a few proposals of practical CNN implementations have been published [3 - 5] and, as a matter of fact, the 8x8 connected-component detector (CCD) of [5] represents the only CNN chip fabricated and tested so far.

In practice, the possibility to manage a neural network which can be used for different applications in different time periods of the signal processing procedure is particularly interesting, as one can use the same part of the chip to perform several circuit functions. In this context, a CMOS approach for the design of a reconfigurable cellular network in which the template coefficients can be digitally varied has been proposed in [4].

In this work we will present the design of an analog programmable CNN architecture with low-power dissipation in a 1.5um CMOS technology.

In particular, after discussing the design of basic building blocks, we will report the electrical performances of a 10x10 CMOS CNN, constituted of about 8,000 MOS transistors, fully simulated at the device level, which can l. :mply analog-programmed by varying an external control voltage. By so doing, the designed CNN can perform such function as noise removal, hole filler, shadow detector, connected component recognition and edge detector. The whole power consumption of the circuit is limited to about 60mW, which is about 1/3 compared to the power consumption of the non programmable circuit presented in [3].

A programmable CNN architecture can be used for several applications in image processing. For example, hand-written character recognition is known to be very sensitive to preprocessing, which differs for constrained and unconstrained writing, moreover, for the single classifiers on cooperating recognition systems. A programmable CNN architecture used for the normalization stage of the classifier allows to apply the same hardware to the above cases. The image size is usually 32x32 or less and requires the processing of 200÷2000 characters per second, making the CNN architecture a very good candidate for this purpose.

2 DESIGN OF THE CMOS BASIC BUILDING BLOCKS FOR THE CNN

The processing unit (N-shaped resistor) was achieved by feedback connecting an inverting transconductance CMOS operational amplifier. The schematic diagram of the circuit is shown in Fig. 1, while Fig. 2 reports the simulated I(V) characteristics of the designed circuit. As it can be seen, the current and voltage values corresponding to the two knee points in the curve are symmetric and their value are 1uA and 0.6V, respectively.

The linear voltage-dependent current sources required to realize the coefficients for template B were realized with the simple OTA circuit shown in Fig. 3, which has the inverting input always connected to ground. Fig. 4 reports the simulated characteristic of the circuit for a design chosen to give a weight of 2. For a ±5V bias voltage the two above circuits have a power consumption of about 75uW and 55uW, respectively.

The circuit used for analog programming the coefficients for feedback template A is shown in Fig. 5 and was developed with the aim of reducing at most the number of control lines, the number of MOS devices, the active gate area and the power consumption. It is constituted by a 6 transistor multiplier cell, a current mirror M1-M2 which establishes the current of the main circuit, a simple output mirror M9, M14 and a double output current mirror M10-M13.

In particular, the drain current Id2 of device M2 is selectively diverted trough M3 or M4 depending on the difference between the input control voltage Vc and the reference voltage Vbias. The partition of Id2 trough M3 and M4 constitutes the programming action of the control voltage Vc. The circuit enables to get for the lateral coefficients of the template A values which can be continuously varied from 8 down to -8.

Fig. 6a shows a typical behaviour of the saturation output current Ios as a function of Vc, while Fig. 6b shows the output current Io as a function of the state voltage Vx of the cell, for several values of A ranging from -0.25 to -8.

3 APPLICATIONS

By using the building blocks previously described we designed several 10x10 CNNs with the aim of validating the design of the programmable A generator. The parasitics effects associated to overlap capacitances and junction capacitances, which vary by changing transistor dimensions and bias voltages, were accurately taken into account. The resulting circuits have a complexity of the order of 8000 transistors and were completely simulated at the device level by using the circuit analysis program Spice. The state capacitor was 1pF.

3.1 CNN Low Pass Filter for Noise Removal

In this application (LP-CNN) the image was continuously applied to the input and applied as initial condition to the state capacitors. The feedforward template B and the "first choice" for the feedback template A were the same of Ref. [3].

The designed LP-CNN was tested by processing several input images with both different voltage levels corresponding to the black and white colours and different superimposed gaussian noise. For black and white levels shrinked together down to 60% of the dynamic range and a gaussian noise with a standard deviations not higher than 0.225 the designed LP-CNN perfectly removes the input noise. Conversely, when the dynamic range of the input signal was made smaller (40% of the dynamic range) and the standard deviation of the noise was increased (s = 0.75) the LP-CNN shows errors in noise removal.

For an image with s = 0.75 an example is shown in Fig. 7, where the output image found by the LP-CNN after noise removal (b) and the correct image (a) are reported. There are 7 pixels containing an error in the output image (a similar result was achieved in [3]). The time required from the LP-CNN to reach the equilibrium state is less than 2usec.

Due to the programmability of the Aij coefficients, several other analyses were performed by continuously varying the value of the "lateral" weights in the A matrix (which were 1 in the original case) from 1.5 down to 0. We found that by increasing Aij the final values of the state voltage slightly change, but the number of errors Ne in the output image does not change. Conversely, when the values of the lateral Aij's decrease gradually from 1.0 down to 0.25, we found more pronounced variations in the final values of the state output voltages and also a sensible decrease in the number of errors. The above small sensitivity of the output image to variations in the values of the lateral Aij weights seems indicate that: a) the "preprocessing" filtering action given by the template B, which is permanently applied on the input image, is very strong and, b), the template A seems just to establish the transient behaviour toward well defined final state voltages.

These observations were further validated from the results achieved by programming the Aij's equal to zero: in such a case when the LP-CNN just does a dynamic filtering action in which the state of each cell is independent on the evolution of the state voltage of the neighbouring cells, we do not found a variation in Ne.

3.2 CNN for Edge detection

For this application we chose for the templates A and B the same values reported on Fig. 15 of [2]. As a processing example we used the 10x10 diamond structure of Fig. 8a. The result achieved by programming the template coefficient values are shown in Fig. 8b. As can be seen, results equal to those found by using the theoretical model of [1] have been obtained. Additionally, results equal to those found in [2] by varying the template coefficients were found.

3.3 Other examples

The designed CNN was also programmed to operate other functions useful for image processing such [6 - 9] as hole filler, connected-component detector and shadow detector. In all the examined cases the circuit simulation indicates a correct operation of the CNN.

REFERENCES

1. L. O. Chua et al., IEEE Trans. on CAS, vol. 35, N. 10, 1988.
2. L. O. Chua et al., IEEE Trans. on CAS , vol. 35, N. 10, 1988.
3. J. E. Varrientos et al., IEEE Int. Workshop on CNN, Budapest, Dec. 1990.
4. K. Halonen et al., IEEE Int. Workshop on CNN, Budapest, Dec. 1990.
5. J. M. Cruz et al., IEEE Trans. on CAS, vol. 38, N. 7, 1991.
6. L. O. Chua et al., IEEE Trans. on CAS, vol. 38, N. 11, 1991.
7. T. Matsumoto et al., IEEE Trans. on CAS, vol. 37, N. 5, 1991.
8. T. Matsumoto et al., IEEE Trans. on CAS, vol. 37, N. 5, 1991.
9. T. Matsumoto et al., IEEE Trans. on CAS, vol. 37, N. 5, 1991.

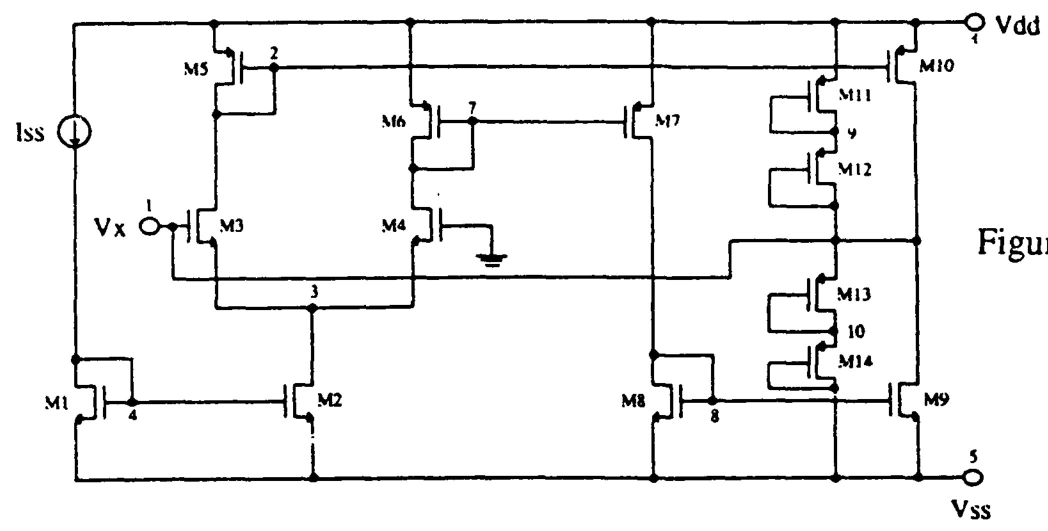


Figure 1

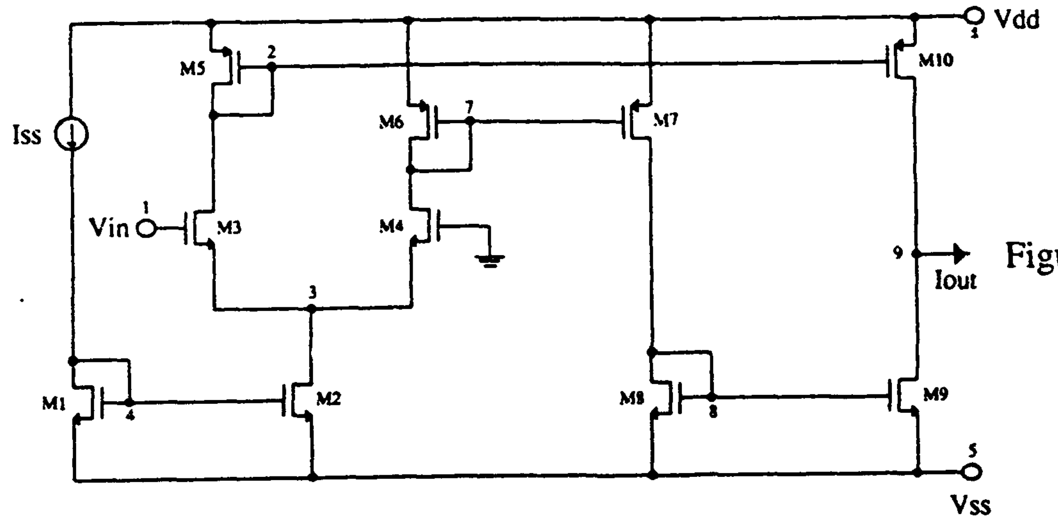


Figure 3

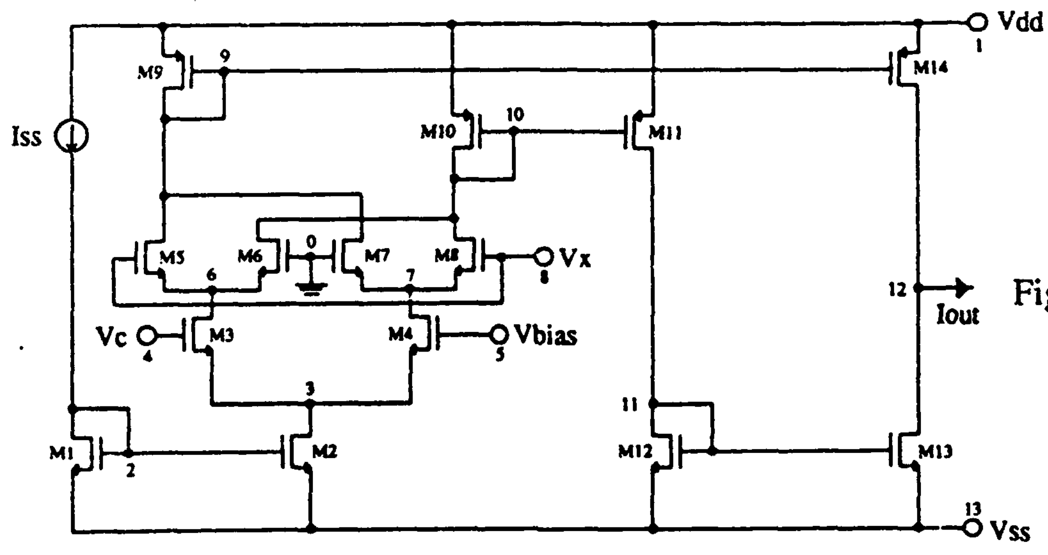


Figure 5

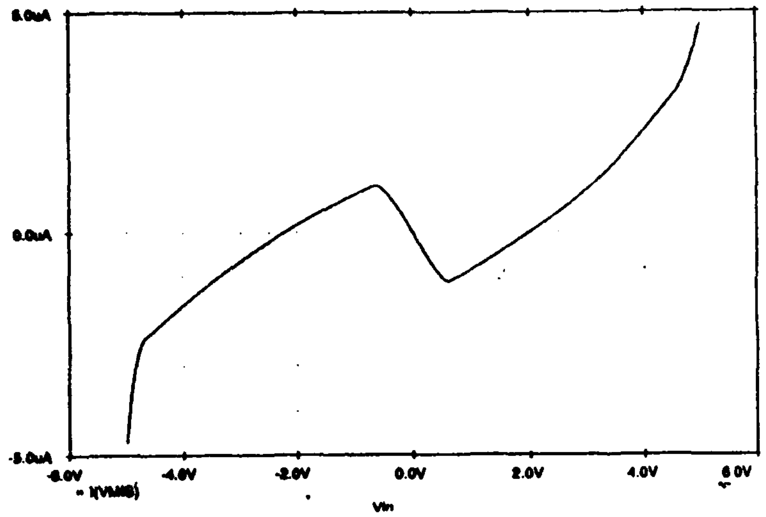


Figure 2

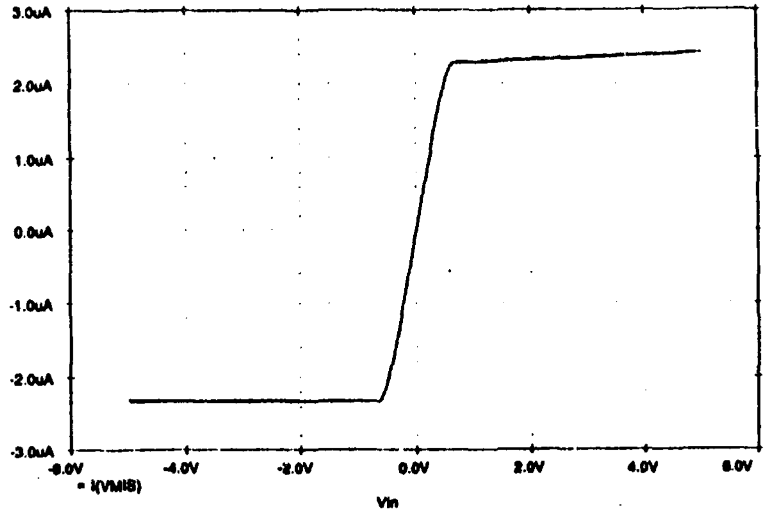


Figure 4

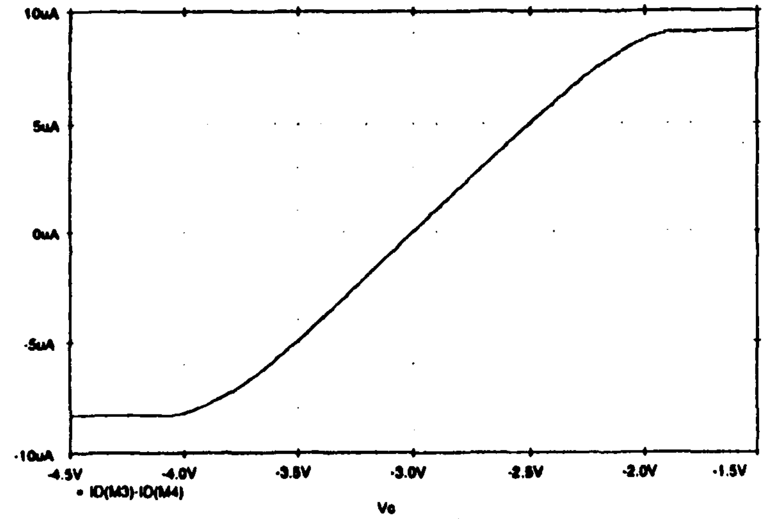


Figure 6a

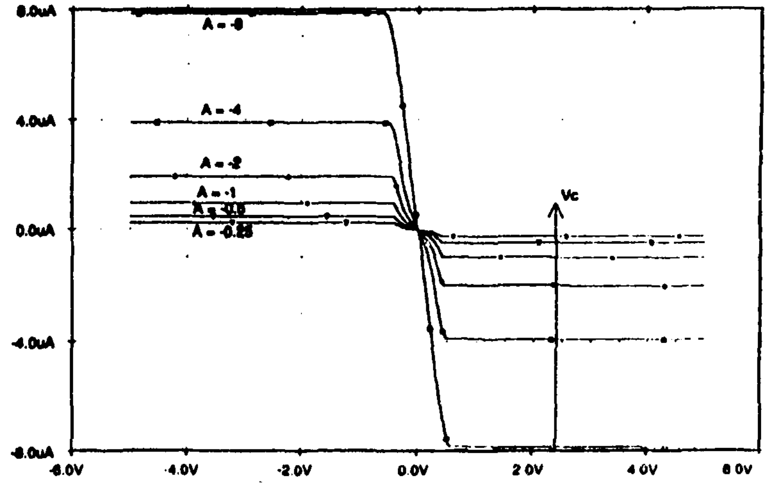


Figure 6b

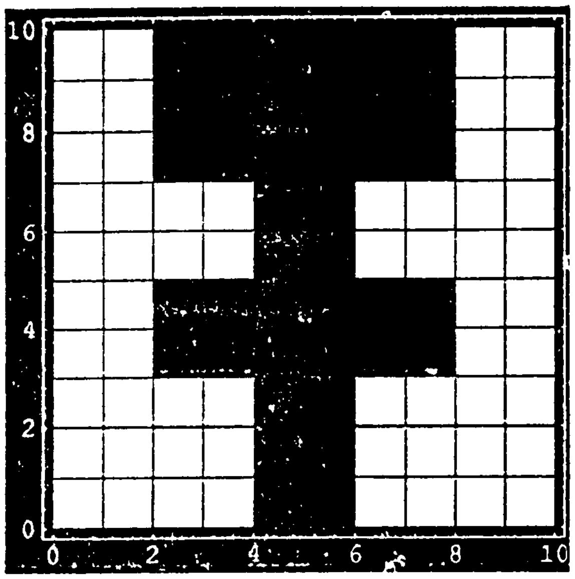


Figure 7a

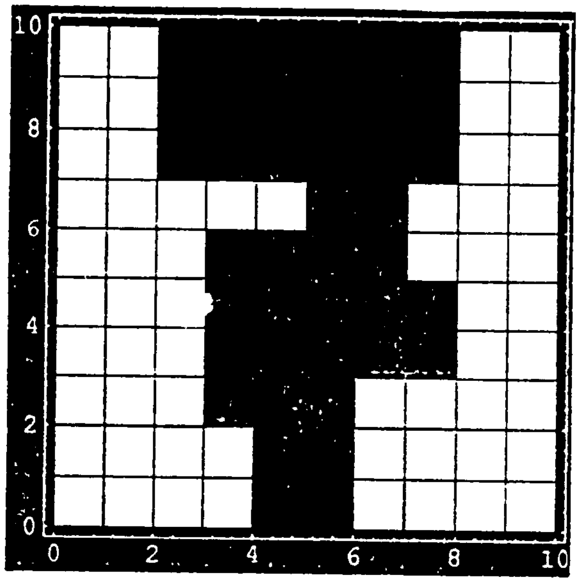


Figure 7b

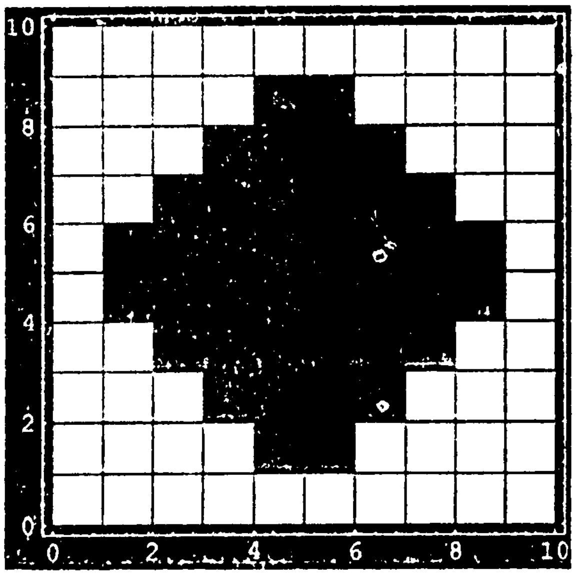


Figure 8a

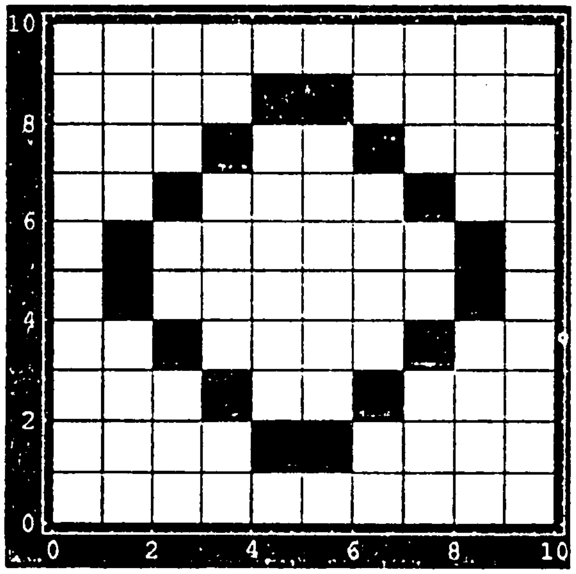


Figure 8b

Multiple-input OTA Based Circuit for Cellular Neural Network Implementation in VLSI CMOS Technology

Tomasz Kacprzak and Krzysztof Ślot
Technical University of Łódź
Institute of Electronics
Stefanowskiego Str. 18/22, 90-924 Łódź, Poland
Ph.(+48 42) 312623, fax.(+48 42) 362238

Abstract

An operational transconductance amplifier with multiple inputs, each of them realized by means of two MOSFETs only, suitable for implementation of Cellular Neural Networks in CMOS technology is described. Preliminary analysis of short-channel circuit augmented by SPICE simulation is given. Results show potential ability of designing chip area effective networks applicable to various tasks of image processing and pattern recognition in real-time. This needs extention of the research especially in the aspect of VLSI implementation.

1. Introduction

Even though the Cellular Neural Networks (CNN) as originally proposed in [1,2] are expected to be very efficient in VLSI implementation the research on this subject has not succeded so far in establishing standard circuit realization of a single cell and architecture of the network as a whole. Structures of the cell circuits proposed in [3-6] are still far from solution which can satisfy CNN requirements and more study augmented by computer simulation and verification with VLSI test chips are highly recommended. The first succesfully constructed chip of CNN connected component detector [7] is very encouraging sign and stimulates the research in the hope of achieving significant progress in this area.

Main question in the design of CNN cell circuit in VLSI technology is how to solve efficiently the problem of controlling the cell by signals coupled in from itself and other cells of given neighborhood in the network. In this sense efficiently means realizability taking into account restrictions of the VLSI design rules, minimization of the chip area and power consumption as well as flexibility in implementation of dynamical processing algorithms suitable for specific applications of CNNs.

List of design requirements for VLSI CNNs includes the following technological problems:

1. Realizability of feedback operator A (cloning template), control (feedforward) operator B and polarization vector I. This requirement is limited by wiring method and floorplanning of the VLS layout. Due to this limitations not every CNN can be realized in a given technology.
2. Programmability of operators A, B and I, defined as the ability to alter the values of the weights between cells in a given neighborhood and values of the polarization vector I. This ability extends CNN tasks that can be realized in image processing and pattern recognition.
3. Method of inputing to and outputing from the CNN the visual information to be transformed.

4. Compatibility with digital microprocesor systems which enables connection of CNNs as a extending cards for speeding up calculations.
5. An influence of nonidealities and tolerances of semiconductor elements as well as frequency-dependent loss, crosstalk and immunity to electromagnetic interference for stability of the CNNs.

Taking into account the main requirements as stated above we present main results of our preliminary study of potential applications the concept of multiple-input OTA circuit in CMOS technology [8]. We concentrate only on the problem of designing the main block of CNN cell circuit, namely the block of spatial convolution the outputs from neighborhood cells. Referring to the nonlinear differential equation [2]:

$$C_X \dot{V}_x^{\,c} = - V_x^{\,c} / R_x + A^c_d V_y^{\,d} + B^c_d V_u^{\,d} + I^c \qquad (1)$$

which describes the dynamics of the cell, this block realizes summation of the three last components of the rigth-hand side of (1) or equivalently the summation at node N currents comming from voltage-controlled current sources, Fig.1.

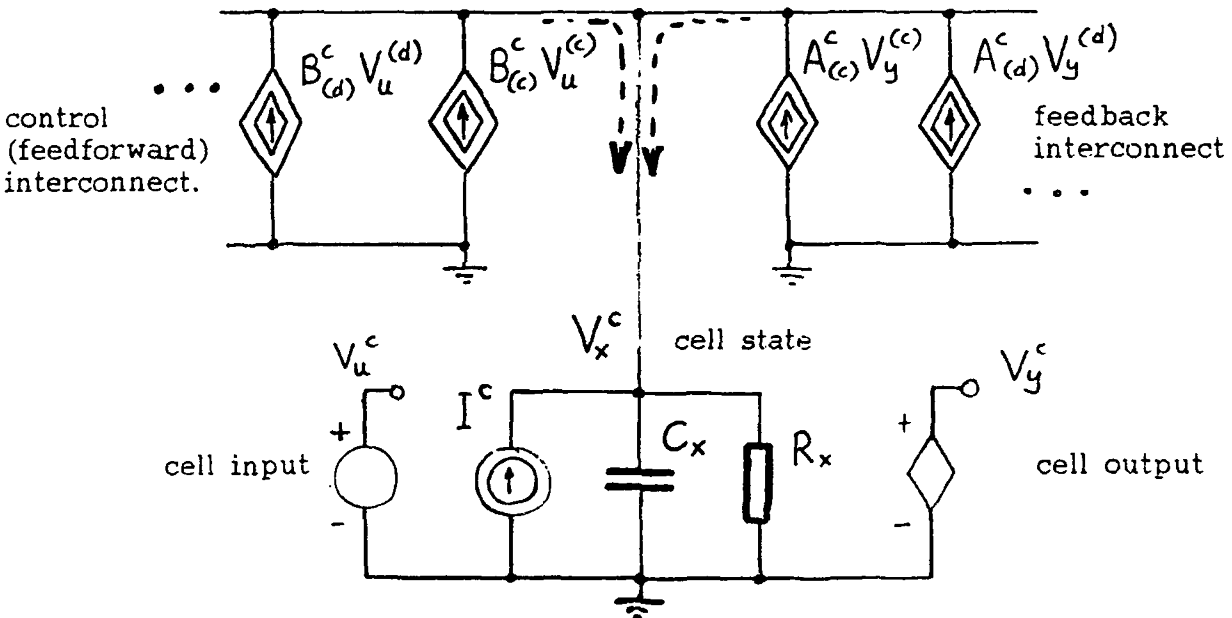


Fig.1. CNN cell circuit of the cell labeled by c. By dotted lines signals caming from neighboring cells are drawn.

Results obtained during this work empower us to develop this idea much more extensively, especially in the aspect of optimization of the VLSI cell and programmability of the network in real-time image processing systems.

2. Multiple-input OTA circuit for CNN cell

The fundamental circuit of multiple-input OTA as proposed in [8] is shown in Fig.2. The OTA is driven by two-transistors input structures (M_{kL} - M_{kU}) connected in parallel to both sides of current mirror M_1 - M_2. The circuit output current I_o is equal to the difference in node currents I_a and

I_b forced by left-hand and rigth-hand sides of input structures.

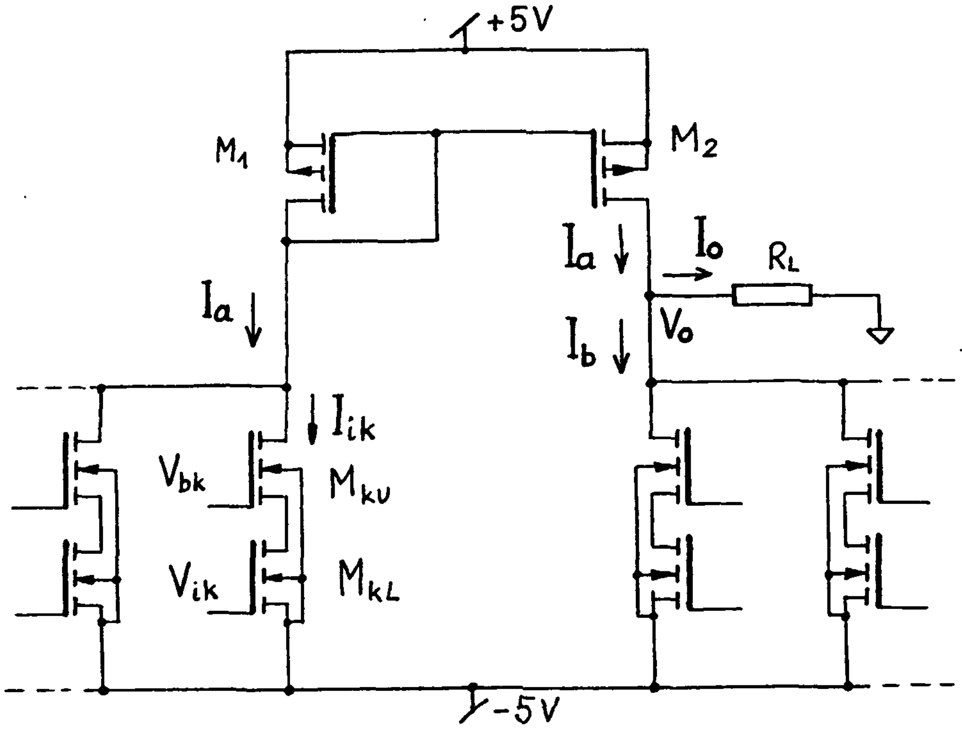


Fig.2. Multiple-input OTA circuit [8]

Each two-transistors input structure, composed of lower M_{kL} and upper M_{kU} devices, provides its own branch current I_{ik} dependent on input voltage V_{ik} and bias voltage V_{bk} according to the formula, derived from square-law MOSFET model:

$$I_{ik} = \beta_k (V_{bk} - V_T) [V_{ik} - 0.5 (V_{bk} + V_T)] \qquad (2)$$

where $\beta_k = \mu$ Cox W/L is determined by the fabrication process and the size of k-th lower transistor M_{kL} and V_T is its treshold voltage. The equation (4) is valid only in this case when the lower transistor operates in its active (nonsaturated) region and the upper transistor - in its saturation, and while the geometrical aspect ratio W/L of upper device is much greater then the corresponding aspect ratio of lower device. It seen from (2) that any input structure can be considered as a transconductance element controlled by input voltage V_i of this element. The transconductance can be altered by the transistor size W/L and/or by the bias voltage V_b. If necessary, the offset voltage 0.5 $(V_{bk}+ V_T)$ contributed in a branch can be easily compensated by the similar branch placed in opposite side of the OTA circuit. Also, the offset component can be used for realization the vector I^c. Therefore, using these input structures one can realize algebraic summation of feedback $(A^c_d\ V_y^d)$ and feedforward $(B^c_d\ V_u^d)$ signals of CNN cell as well as its bias current (I^c).

Computer simulation shows that the error introduced in the approximation (2) is negligible when the aspect ratio $(W/L)_U$ is at least 200 times greater then the aspect ratio $(W/L)_L$. Assuming 2 μm CMOS feature size this creates large chip area occupied by single input and correspondingly very large area by the whole cell (one cell can be controlled by 16 current sources even in the case of minimal neighborhood radius).

To safe chip area the MOSFET technology with channel length down to 1 μm or even smaller should be used. However, for such small channel length new transistor model with carriers drift velocity effect must be considered. The use of the long-channel square-law model for the short-channel devices gives considerable discrepancies in both the dc and transient solutions between the calculated and measured results and can lead to incorrect conclusions in regard to circuit performance. It was shown [9], that when the power supply of +5V and -5V is used (which is typical standard supply in digital-analog integrated circuits) then the MOS transistor model in saturation for short-channel device can be described by:

$$I_D = v_{sat} C_{ox} W (V_{GS} - V_T) \qquad (3)$$

where v_{sat} is carrier drift velocity saturation equal to approximately $7x10^6$ cm/s for both the NMOS and PMOS devices. As was expected in the theory of short-channel devices, the drain current in saturation region of I-V output characteristics does not depend on the channel length. Also, the bulk effect is negligible and the output conductance of the device is included through the threshold voltage dependent on drain-source voltage.

The I-V characteristic of the k-th two transistor structure (Fig.3) with short-channel devices can be derived by equating the drain current of the lower transistor M_{kL} operating in nonsaturation with the drain current of the upper transistor M_{kU} operating in saturation region. After all stages of this derivation one can obtain the following formula for the output current I_k:

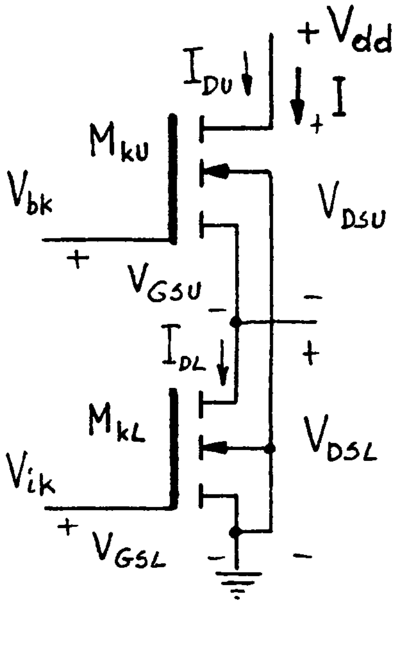
$$I_k = \beta_k(V_{bk} -V_T) [V_{ik}-V_T- 0.5 (V_{bk}-V_T)] [1 + a(V_{ik}-V_T)] \qquad (4)$$

where the constant a is expressed by (L_k is the channel length of the lower transistor):

$$a = \mu_o W_L /(v_{sat} W_U L_k) \qquad (5)$$

Assuming transistors geometry: $(W/L)_U$ = (10μm/1μm) and $(W/L)_L$ = (1μm/10μm) and the value of electrons mobility μ_o = 500 cm^2/V.s, constant a gets the value a = 0.007 V^{-1}. Therefore, for the voltage range -5V to +5V the following enequality holds: 1 $a(V_{ik}-V_T)$. In this case the formula (4) for the output current of the transconductance element with short-channel devices is equal to the corresponding formula (2) valid for long-channel devices, but with considerable reducing of area uccupied by the element.

Fig.3 shows dc characteristics of the transconductance element. It can be noted that for V_i in the range between +5 and +10V the element behaves like a linear transconductor with the error of maximum value not exceeding 20% . Since neural networks do not require a very linear response, this error can be quite well acceptable for many CNN applications.



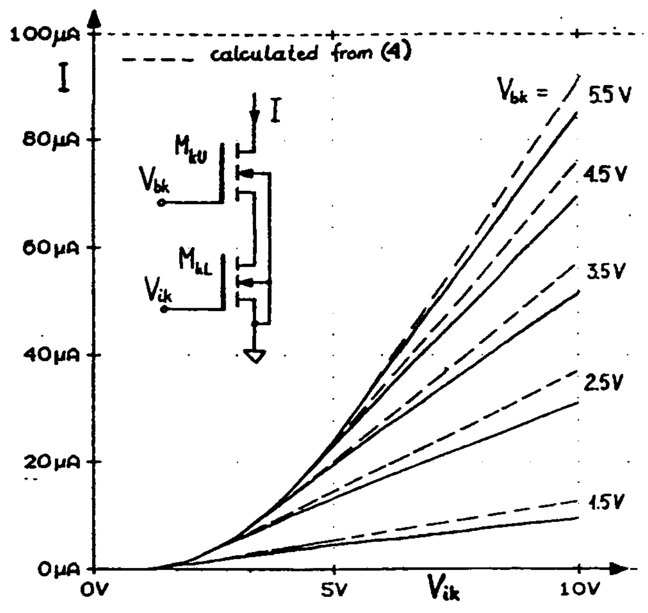
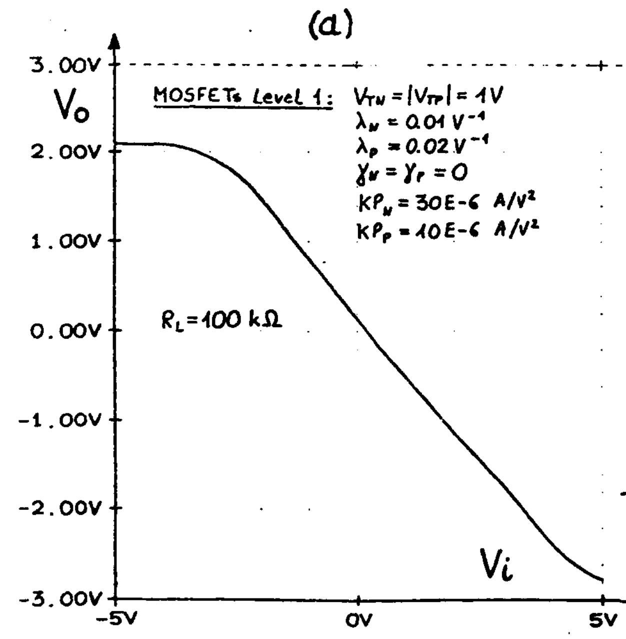


Fig.3. Two-transistor transconductance element with short-channel devices (a) and its DC transfer characteristics



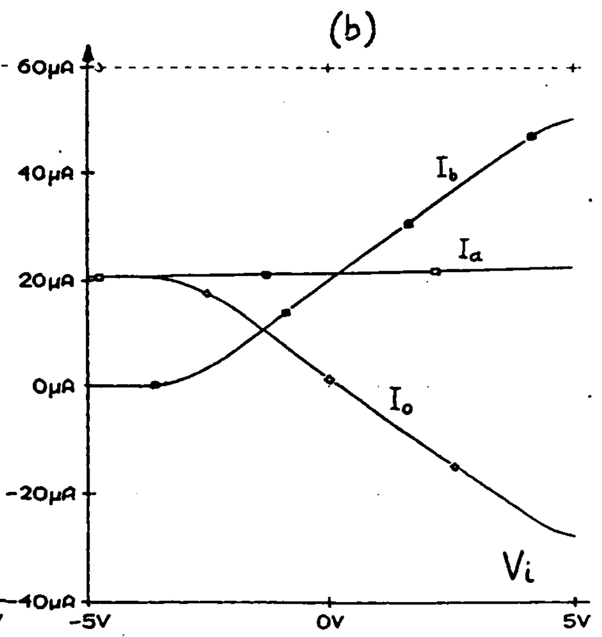


Fig.4. (a) - dc voltage transfer characteristics, (b) - dc current transfer characteristics for two-inputs OTA circuit of Fig.2. Results were obtained using SPICE simulation with MOSFET model Level 1

Fig.4. shows simulation results of DC voltage and current transfer characteristic of two-input OTA. It is seen from this figure that the linear range of the output voltage spans of several volts while controlling the input almost from minimal to maximal voltage of power supply. This output voltage range is also quite well acceptable for most applications of CNNs.

3. Conclusions

A multiple-input OTA circuit as proposed in [8] has been adopted in preliminary analysis of VLSI CMOS implementation of Cellular Neural Networks. The circuit generates an output current which is a linear function of sum of the feedback and feedforward signals coming from neighbour cells. The circuit enables to alter the transconductance value of each voltage-controlled current source by means of changing the bias voltage of upper MOSFET device, as well as by means of changing the aspect ratio (channel width to channel length) of the lower (controlled) device. This makes good flexibility in the design of CNN cells and the whole circuit for given image processing and/or pattern recognition applications. SPICE simulation and hand analysis of the circuit with short-channel devices show quite good linearity with economical occupation of chip area. Future work should be concentrated on simulations dynamical characteristics and on design procedure for specific CNN applications.

ACKNOWLEDGEMENTS: Authors would like to thank Professor L.O.Chua from The University of Berkeley (USA) and Professor T.Roska from Hungarian Academy of Science for their encouragements to undertake research on CNN.
This work was supported partly by The State Committee for Scientific Research under Grant No 3 0611 91 01.

REFERENCES:

[1] L.O.Chua and L.Yang, "Cellular Neural Networks: Theory" and "Cellular Neural Networks: Applications", IEEE Trans. on Circuits and Systems, CAS-35, 1988, 1257-1290.

[2] J.A.Nossek, G.Seiler, T.Roska and L.O.Chua, "Cellular Neural Networks:Theory and Circuit Design", Report No. TUM-LNS-TR-90-7, Technische Universitat Munchen, 12 Dec. 1990.

[3] T.Kacprzak and K.Slot, "Problems of Cellular Neural Networks Implementation in CMOS Technology", Proc.XIII National Conf. Circuit Theory and Electronic Circuits, Bielsko-Biala (Poland), Oct. 1990, 355-360.

[4] K.Halonen, V.Porra, T.Roska and L.O.Chua, "VLSI Implementation of Reconfigurable Cellular Neural Network Containing Local Logic (CNNL), Proc. IEEE CNNA-90, Budapest (Hungary), 206-215.

[5] -ibid, J.E.Varrientos, J.Ramirez-Angulo and E.Sanchez-Sinencio, "Cellular Neural Network Implementation: a Current Mode Approach", 216-223.

[6] -ibid, A.Rodriguez-Vazquez, R.Dominguez-Castro and J.L.Huertas, "Accurate Design of Analog CNN in CMOS Digital Technologies", 273-279.

[7] J.M.Cruz and L.O.Chua, "A CNN Chip for Connected Component Detection", IEEE Trans. Circuits and Systems, CAS-38, 1991, 812-817.

[8] R.D.Reed and R.L.Geiger, "A Multiple-Input OTA Circuit for Neural Networks", IEEE Trans. Circuits and Systems, CAS-36, 1989, 767-770.

[9] K-Y Toh, P-K Ko and R.G.Meyer, "An Engineering Model for Short-Channel MOS Devices", IEEE J. Solid-State Circuits, Vol.23, 1988, 950-958

New Test Results of a 4 by 4 Discrete-Time Cellular Neural Network Chip

by Hubert Harrer and J. A. Nossek

Technical University Munich, Inst. f. Network Theory and Circuit Design
Arcisstr. 21, D-800 Munich 2, Germany

Abstract

In [5] an analog implementation of discrete-time cellular neural networks is described that is based on the idea of conductance multipliers [1], [10], [7] and transconductance amplifiers [8], [11]. The circuit architecture takes advantage of differential currents suppressing dc disturbances. The paper gives new measurement results that compare the fabrication tolerances between different cells on the chip and the offset of the single cells.

1 Introduction

Cellular Neural Networks [2] have been shown to be an efficient tool in image processing. However, the whole capability of the system is only exploited if specific hardware realizations are provided. Analog VLSI implementations have the advantage that the CMOS transistors are not only used as switches leading to a very small number of devices and therefore low chip area. On the other hand analog circuits are much more sensitive to disturbances and can only be applied if the architecture is robust enough. For cellular neural networks analog proposals are given in [4], [3]. In [9] a circuit is described realizing a discrete-time implementation of this architecture. The circuits differ in programmable and fixed template coefficients or in an additional local logic for the output states.

Discrete-time cellular neural networks (DTCNN) [6] are derived from CNN and the discrete Hopfield model. The network is described by the recursive algorithm

$$x^c(k) = \sum_{d \in N_r(c)} a_d^c y^d(k) + \sum_{d \in N_r(c)} b_d^c u^d + i^c \tag{1}$$

$$y^c(k) = f(x^c(k-1)) = \begin{cases} 1 & for\ x^c(k-1) \geq 0 \\ -1 & for\ x^c(k-1) < 0. \end{cases} \tag{2}$$

if time-invariant templates and inputs are assumed. The variables and coefficients denote:

$x^c(k)$:	cell state	a_d^c:	feedback coefficient	$N_r(c)$:	r-neighbourhood
$y^c(k)$:	cell output	b_d^c:	control coefficient		
u^c:	cell input	i^c:	threshold		

0-7803-0875-1/92 3$.00 ©1992IEEE

The cells are only locally connected within the r-neighbourhood [2], which is defined as the set of all cells within the distance r including cell c. In contrast to the CNN, the nonlinearity is a threshold function and the system is clocked. The constant part can be summarized to the cell bias

$$k^c = k^c(\mathbf{u}) = \sum_{d \in N_r(c)} b^c{}_d u^d + i^c. \tag{3}$$

2 Circuit Architecture

An analog circuit is discussed in [5], [6] for realizing the DTCNN. It implements the feedback part with a 1-neighbourhood on a hexagonal grid and programmable weights. Fig. 1 gives the circuit structure for a single cell realizing the algorithm (1), (2).

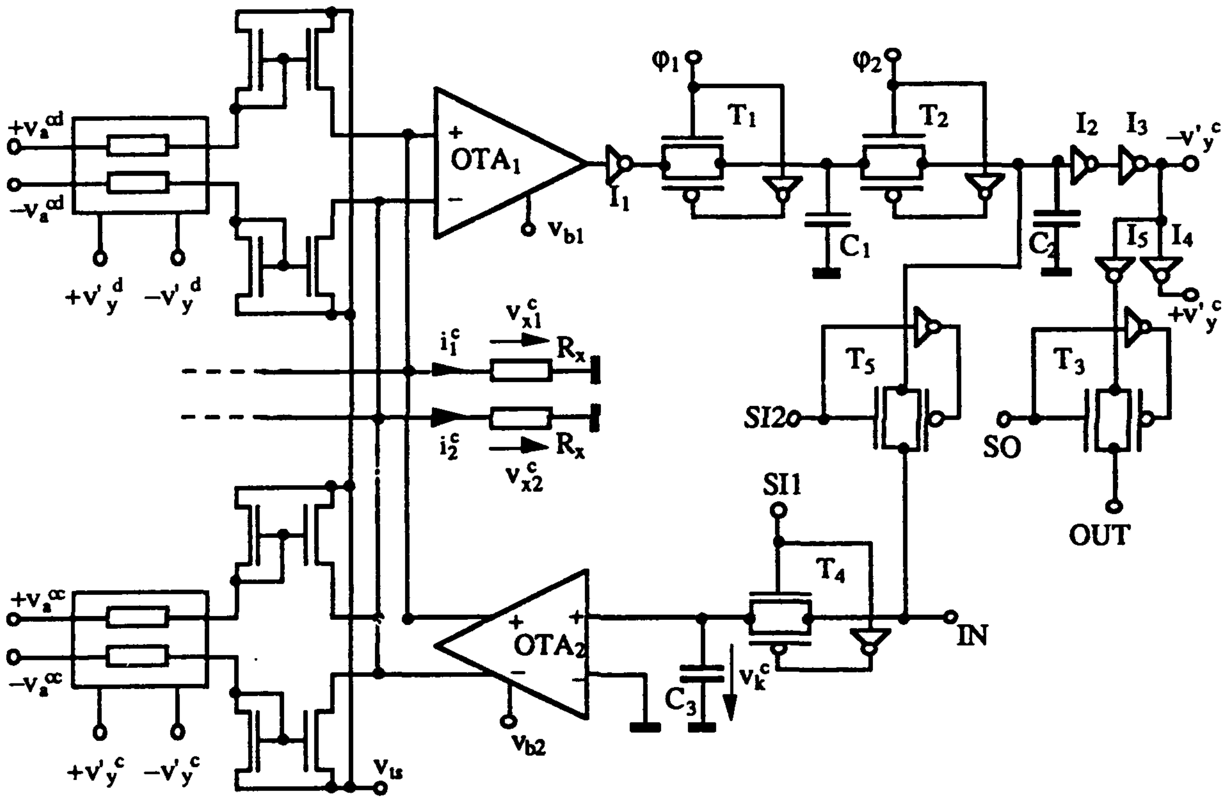


Figure 1: Feedback part realization of a single cell.

The operational transconductance amplifier OTA_2 is used within the linear range to provide a differential output current that is proportional to the input voltage v_k^c, which corresponds to the constant cell bias k^c. The capacitor C_3 realizes an analog memory and is charged via the bus IN, which is activated by SI1. This enables data transfer by the matrix concept, where a whole column is read in parallel. The feedback of the binary outputs $v'^c_y = v^c_y + V_T$ is achieved by the conductance multipliers, which are controlled by the global weight voltages $\pm v_a^{cd}$. Here, V_T denotes the threshold voltage. Their output currents are summed and transformed into a voltage v_{x1}^c and v_{x2}^c. Then a difference voltage

$$v_x^c(kT) = v_{x1}^c(kT) - v_{x2}^c(kT) = R_x(i_1^c(kT) - i_2^c(kT)) = R_x\left(\sum_{d\in N_r(c)} A_d^c v_y^d((k-1)T) + g_m v_k^c\right) \tag{4}$$

is obtained , where g_m denotes the conductance of the OTA_2 and

$$A_d^c = 2\frac{W}{L}\mu C_{ox} v_a^{cd}. \tag{5}$$

The parameters W and L define the transistor geometry; μ is the mobility and C_{ox} the oxide capacitance per unit area.

The OTA_1 is operated as a comparator and decides the sign of the state voltage $v_x^c = v_{x1}^c - v_{x2}^c$. The system is switched into the next output state by the dual nonoverlapping clock signals φ_1 and φ_2. Data are read out via the bus OUT when activating the transmissiongate T_3. The initial values are loaded by the bus IN using the transmission gate T_5. For a detailed analysis refer to [5], [6]. A 4 by 4 cell chip has been designed for experimental purposes using this circuit architecture.

3 Measurement Results

In CMOS processes fabrication tolerances may occur, which are related due to small changes of the oxide thickness, different donor concentrations and a possible fringing of structures. This may cause different threshold voltages for the transistors leading to deviations of the saturation currents. Some of the effects can be suppressed by a specific geometrical design in the layout, but in general they cannot be excluded. This section gives static and dynamic measurement results of the test chip described in Section 2. It has been fabricated on a standard digital 1.5 μ single poly double metal process at ES2.

If all weight voltages $\pm v_a^{cd}$ and the input voltage v_k^c are set to zero, a constant cell offset has been observed, which is given exemplarily in Table 1 for a single chip.

cell 1:		cell 5:	0.21 V	cell 9:	0.11 V	cell 13:	−0.04 V
cell 2:	0.04 V	cell 6:	0.13 V	cell 10:	0.04 V	cell 14:	
cell 3:		cell 7:	0.02 V	cell 11:	0.16 V	cell 15:	0.11 V
cell 4:	0.00 V	cell 8:	0.18 V	cell 12:	0.02 V	cell 16:	0.18 V

Table 1: Offset voltage of the cells.

Cell 1, 3 and 14 have no measurement bus for the state voltages v_{x1}^c and v_{x2}^c and, therefore, their offset cannot be determined directly. The cell offset is caused by a mismatch of the differential currents of each circuit component, which is summed up in the worst case. Compared with the maximum linear range of ±2 V of the transconductance amplifiers OTA_2, this offset seems to be very large. However, it is a constant value and does not change during operation. Hence, it can be compensated with the input voltage v_k^c of the OTA_2. This makes a complex compensation circuit superfluous.

Fig. 2 shows the transfer characteristic of the offset compensated OTA_2 for 13 cells, if all weight voltages are set to zero. The bias voltage of the OTA_2 has been chosen to $v_{b2} = -1.35$ V and v_{ts} was set to -2.0 V. The power supply amounts to $v_{dd} = 2.5$ V and $v_{ss} = -2.5$ V.

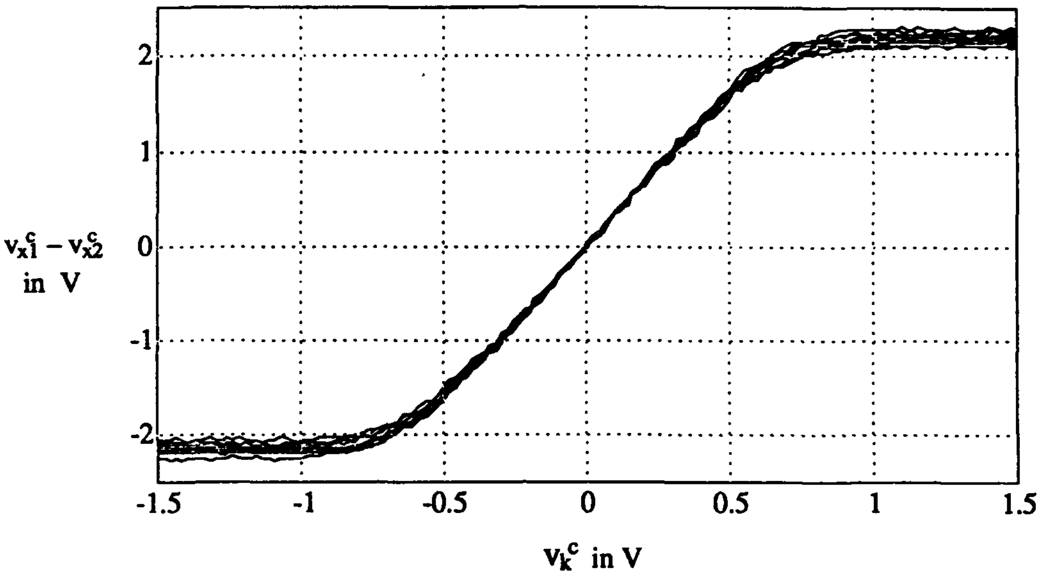


Figure 2: Dc transfer characteristic of the OTA_2 for 13 cells.

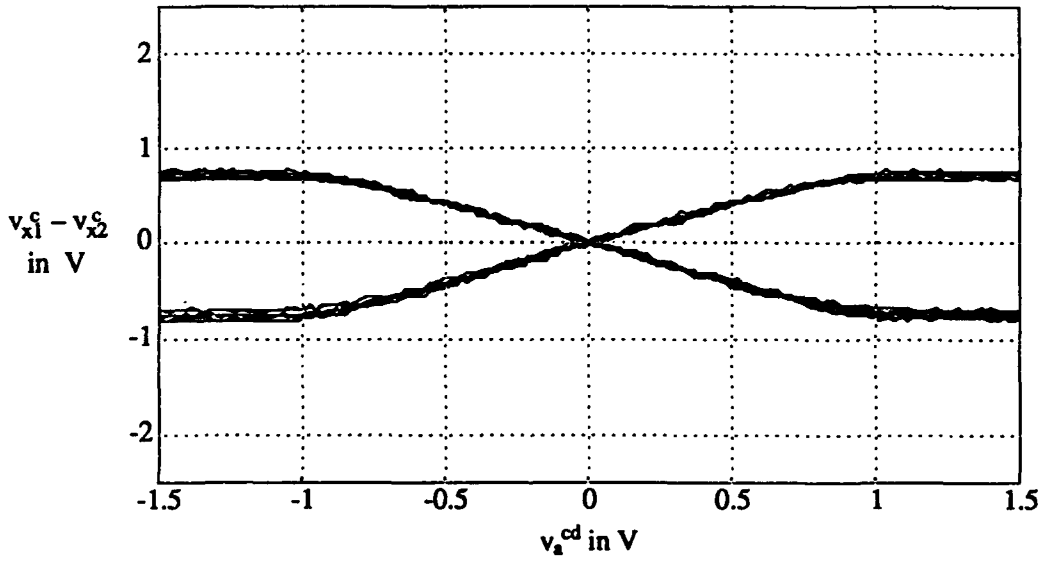


Figure 3: Dc transfer characteristic of a single conductance multiplier for 13 cells.

Since the saturation voltages are slightly different, their slopes have identical values within a linear range, which stretches between -0.5 V and 0.5 V. The slope of the OTA_2 can be adjusted by the bias voltage v_{b2}. Then, the saturation voltage is shifted between 0 V and 2 V, too, while the linear range remains unchanged. It corresponds to a gain between 0 and 3.0 of the input voltage. Because the resistors R_x have been designed to a resistance of 200 kΩ, a maximum transconductance $g_m = 15$ μS has been obtained for the OTA_2.

In Fig. 3 the dc transfer characteristic of a single weight coefficient is illustrated for 13 cells. The remaining feedback coefficients are set to zero and the input voltage v_k^c compensates the offset.

The linear range stretches between −0.9 V and 0.9 V with a gain of 0.8. This corresponds to a maximum absolute value of 5 μS for A_d^c, when inserting the process and design parameters in (5). No significant deviations have been observed for different weight coefficients. This is also true for a comparison of the characteristics between different chips.

The network variables and parameters are retransformed by

$$a_d^c = A_d^c R_x, \quad x^c(k) = \frac{v_x^c(kT)}{V_{sat}}, \quad y^c(k) = \frac{v_y^c(kT)}{V_{sat}}, \quad k^c = \frac{v_k^c g_m R_x}{V_{sat}} \tag{6}$$

to obtain the algorithm (1), (2). The constraint $y^c(k) = \pm 1$ implies $V_{sat} = | v_y^c(kT) | = v'^c_y - V_T = 0.8V$. This leads to $k^c \in [-2.5, 2.5]$ and $x^c(k) \in [-3.75, 3.75]$ if a linear range of $\pm 1.5V$ is assumed for v_{x1}^c and v_{x2}^c. For the feedback coefficients, $a_d^c \in [-1, 1]$ is obtained from (5) and (6). Notice that the output voltage v'^c_y can be used for scaling, too. It adjusts the range for k^c and x^c, because it determines the reference voltage V_{sat}. Then, the power supply of the inverters I_3 and I_4 in Fig. 1 consists of a scaling voltage v_{scal}.

As an example for dynamic testing, the connected component detector was chosen with the initial pattern on the left side of Fig. 5. A cell bias was loaded to compensate the constant cell offset and the weight coefficients have been set to the values in Fig. 4.

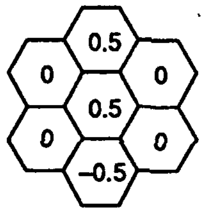


Figure 4: Weight voltage v_a^{cd} of the feedback coefficients in V.

Fig. 5 shows the time-dependent development of the output pattern. After four iterations all outputs are constant. The circuit could be operated even with a clock frequency of 10 MHz. However, for smaller weight coefficients the maximum clock rate goes down.

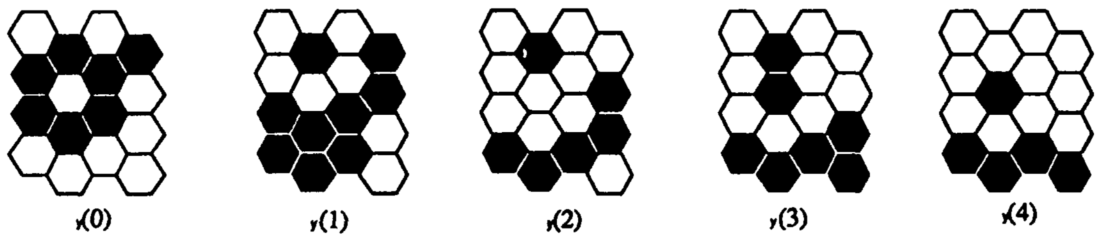


Figure 5: Time-dependent development of the output pattern.

4 Conclusion

The mesurements have shown that the applied architecture was well suited for the realization of discrete-time cellular neural networks. Especially the large computation speed demonstrates the performance of this network structure. Improvements are desirable in the linear range of the transconductance amplifier, because this device needs a large operation range to compensate the output currents of all conductance multipliers as required for many applications. Disturbances in the input voltage imply a large deviation of the state voltage, since they are amplified by the gain of the OTA, which may be much larger than the gain of the multipliers. Hence, the bias voltage of the OTA should be adjusted so that the maximum requested absolute value of the cell bias is yet reached. Further improvement of the circuit properties could be achieved by using cascoded current mirrors. Since their transistor geometries can be chosen very small the size of a cell is not significantly increased.

References

[1] M. Banu and Y. Tsividis, "Fully-Integrated Active-RC Filters in MOS Technology", IEEE Journal of Solid-State Circuits, vol. 18, 644 - 651, 1983.

[2] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory", IEEE Trans. Circuits and Systems, vol. CAS-35, 1257 - 1272, 1988.

[3] J. M. Cruz and L. O. Chua, "A CNN Chip for Connected Component Detection", IEEE Trans. Circuits and Systems, vol. CAS-38, 812 - 817, 1991.

[4] K. Halonen, V. Porra, T. Roska and L. Chua, "VLSI Implementation of a Reconfigurable Cellular Neural Network Containing Local Logic, CNNL", Proceedings of the First IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA-90, 206 - 215, Budapest, December, 1990.

[5] H. Harrer, J. A. Nossek and R. Stelzl, "An Analog Implementation of Discrete-Time Cellular Neural Networks", IEEE Trans. on Neural Networks, vol. 3, 466-477, 1992.

[6] H. Harrer, "Discrete-Time Cellular Neural Networks", Dissertation, Technical University Munich, to appear in 1992.

[7] M. Ismail, S. V. Smith and R. G. Beale, "A New MOSFET-C Universal Filter Structure for VLSI", IEEE Journal of Solid-State Circuits, vol. 23, 183 - 194, 1988.

[8] C. Mead, "Analog VLSI and Neural Systems", Addison Wesley, Reading, 1989.

[9] A Rodriguez-Vazquez, R. Dominguez-Castro and J. L. Huertas, "Accurate Design of Analog CNN in CMOS Digital Technologies", Proceedings of the First IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA-90, 273-280, Budapest, December, 1990.

[10] Y. Tsividis, M. Banu and J. Khoury, "Continuous-Time MOSFET-C Filters in VLSI", IEEE Trans. Circuits and Systems, vol. CAS-33, 125 - 140, 1986.

[11] E. Vittoz, "Analog VLSI Implementation of Neural Networks", Journées D' Électronique 1989, 224 - 250, Lausanne, October, 1989.

Design and Testing Issues in Current-Mode Cellular Neural Networks

S. Espejo, A. Rodríguez-Vázquez and J.L. Huertas
Centro Nacional de Microelectrónica-Universidad de Sevilla
Edificio CICA, C/Tarfia sn, 41012-Sevilla, SPAIN
Tel. 34-5-462 38 11; Fax 34-5-462 45 06
Email: espejo@cnm.us.es, angel@cnm.us.es, huertas@cnm.us.es

Abstract

A general technique for continuous time (CT) and discrete time (DT) Cellular Neural Networks (CNNs) implementation using current-mode techniques is presented. The proposed methodology yields high pixel densities, high speed and low power consumption, while the design procedure is extremely simple. Resulting circuits are well suited for standard digital CMOS processes, since only MOS transistors are required. CNN cell layouts are shown together with electrical simulation results from extracted netlists of complete networks. Montecarlo analysis demonstrates the viability of the proposed techniques. Basic building blocks have been experimentally tested.

Introduction

Cellular Neural Networks are two-dimensional arrays of locally interconnected cells. Each cell in a CNN has three associated variables, namely:

- *Cell state*: $x^c(t)$, which conveys cell energy information as a function of time.
- *Cell output*: $y^c(t)$, which is obtained from the cell state via a nonlinear limitation:

$$y^c(t) = \frac{1}{2}(|x^c(t)+1| - |x^c(t)-1|) = f(x^c(t)) \tag{1}$$

- *Cell input*: u^c, representing external excitation.

The behavior of the network is governed by a system of dynamic equations, one for each cell, which in the original model are given by [Chua88a]:

$$\tau\frac{dx^c(t)}{dt} = -x^c(t) + D^c + \sum_{d \in N_r(c)} \{A_d^c y^d(t) + B_d^c u^d\} \qquad \forall c \in \mathcal{GD} \tag{2}$$

where the coefficients A_d^c and B_d^c, with $d \in N_r(c)$, can be arranged into two matrices, called the *feedback* and *control templates* respectively, and D^c is the *offset* term. The time constant τ is invariant from cell to cell.

The CNN description summarized above corresponds to a CT system. The same input/output mapping can be obtained, under some restrictions, from a DT system obtained using some explicit integration algorithm. The simplest choice is Forward Euler, which results in

$$x^c(n+1) = x^c(n) + \frac{T}{\tau}\left[-x^c(n) + D^c + \sum_{d \in N_r(c)} \{A_d^c y^d(n) + B_d^c u^d\}\right] \qquad \forall c \in \mathcal{GD} \tag{3}$$

0-7803-0875-1/92 3$.00 ©1992IEEE

where n denotes the DT instant and T is the time interval between DT instants. Stability of this system and accuracy in the emulation of the CT model is strongly influenced by the ratio T/τ, which also influences the number of DT instants required for convergency. A good heuristic choice, which has always given correct results in our trials and which significantly simplifies cell design is $T/\tau=1$. With this assumption, (3) can be written as

$$y^c(n+1) = f\left[D^c + \sum_{d \in N_r(c)} \{A_d^c y^d(n) + B_d^c u^d\}\right] \qquad \forall c \in \mathcal{GD} \tag{4}$$

where $f(.)$ is the nonlinear function in (1). Note that state variables do not appear explicitly in (4) and, hence, they are not required. In this case, signal ranges of all variables in the circuit are the same, as opposed to the original CT model in which a higher signal range is needed for state variables [Chua88a].

Practical use of CNNs in any of its potential application fields [Chua88a,b] requires feasible and efficient implementation techniques. Major trends are area and power efficiency, operation speed, testability and ease of communication with the outer world. Current-mode techniques [Toum90] are proposed for CNNs implementation in this paper. The results obtained show better area, power and speed figures than previous proposals [Cruz91, Halo92, Harr92].

Current mode basic building blocks

The operations required for the implementation of (2) and (4) are: *summation*, *weighted replication*, *nonlinear limitation*, *integration* (CT), and one clock-cycle *delay* (DT).

Summation is achieved in current-mode by simply routing different signals to a common node. Weighted replication is obtained by using a simple analog circuit: the *current mirror*. In MOS technologies, the weight is controlled by the sizes of the input and output transistors in the mirror. Since in current mirrors currents always flow in the same direction, bias shifting is required to allow for the symmetric existence-interval of the variables. These concepts are illustrated in Fig.1(a) and (b). Several output currents with the same or different scaling factors can be obtained by using different output transistors.

In Fig.1(b), a saturation nonlinearity appears as a result of the cut-off of the input transistor. By cascading two of these structures, the nonlinearity in (1) is obtained, as shown in Fig.2.

The basic dynamic block for DT systems is obtained from the circuit in Fig.2 by introducing switches in the gate-voltage paths of the current mirrors, as shown in Fig.3. Each switch is driven by one of two nonoverlaped clock signals, yielding a full cycle delay.

Finally, Fig.4 shows a CT lossy integrator which performs according to the following equation

$$\tau \frac{dI_o}{dt} = -I_o + I_{in} \tag{5}$$

where τ is given by $\tau=C/g_m$, g_m being the small signal transconductance of the transistors in the current mirror. Although the time constant τ is a function of the output current (through g_m), it can be shown that this does not affect the dynamic properties of the cell. Bias current in the integrator must be adjusted to provide enough range for state variables. An expression for this range, usually between 5 an 10 times that of the output variables, is given in [Chua88a]. In addition to the integrator, the block that loads it must be capable of sinking (sourcing) currents in the same range.

Cell architecture

Designing CNN cells described by (2) or (4) is straight forward using the blocks described in the previous section. Fig.5 shows a schematic diagram of a cell valid for both approaches.

Weighted replication is achieved with the circuit in Fig.1(b). When sign inversion is required, an additional replication stage is cascaded. In the CT case, the dynamic block consists of a CT integrator loaded with a bias shifted current mirror. Both the integrator and its load must have enough signal range for the state variable. The nonlinear limiter truncates the state variable to the limits of the output variables. In the DT case, the dynamic block consists of the delayer in Fig.3. This block also acts as a nonlinear limiter. Hence, the limiter block in Fig.5 can be eliminated.

For simplicity, I/O circuitry has not been included in the diagram in Fig.5. Initial state values $x^c(0)$ and external inputs u^c can be electrically set, or optically transmitted using photoactive devices [Sayl91]. Output of each cell can be evaluated with the help of an additional replication branch. Further I/O considerations are left aside in this paper due to the limited available space.

Note that weighted replication is performed at the output of each cell. In other words, each cell produces a different output, with the required weight, for each neighbor. At each cell, neighbor contributions are summed at the input node as they are received. In order to avoid confusion when designing current-mode CNNs, it is convenient to work with new template matrices, obtained from the original ones by interchanging entries along all radial lines of the matrices.

To conclude this section, Fig.6 shows two complete examples of cell schematics, one for CT and another for DT. Both are designed for connected component detection (CCD). Note the reduced complexity achieved with the DT approach, due to the common range of state and output variables. This results in significantly better area an power figures. On the other hand, problems associated to DT analog circuits must be taken in account (feedthrough, noise). A new CT CNN model which combines the advantages of the DT approach is described in a separate paper [Huer92].

Results

Several prototypes have been designed following the proposed technique on a 1.6µm single-poly double-metal digital CMOS process. As an example, results of a CT CNN for CCD will be described here. The design actually includes two prototypes, each with sixteen cells in a row. First prototype (P1) corresponds to the schematic in Fig.6(a) while the second one (P2) corresponds to the simplified CT model mentioned above, whose schematic can be viewed as that of Fig.6(b) when both switches are simultaneously *on*. Area and power figures of prototype P2 are essentially the same than for a DT design.

Design process begins by choosing a *unitary bias current* (I_Q) which in our case was 2µA. A current mirror capable of driving $2I_Q$ and a I_Q current source must then be designed. In our prototype, cascode structures were used to implement both the current mirror and the current source, all transistors having w/l=4.0µm/3.2µm. After this steps, implementing any CNN is just a matter of combining the same building blocks.

Power calculation is extremely simple from cell schematic. With a 5V power supply, P1 consumes 290µW/cell, while P2 takes only 90µW/cell. Fig.7(a) and (b) show the layouts of P1 and P2 single cells respectively, including the initialization circuitry, an extra output for evaluation purpose, and the necessary spacing and lines to be connected by abutment with neighboring cells. Area figures are 16.368μm^2/cell (equivalently 61 cells/mm^2) for P1, and 8303μm^2/cell (equivalently 120 cells/mm^2) for P2.

Fig.8 shows simulated electrical results from the layout-extracted netlist of the prototype P2, which is slightly faster than P1. In order to test the prototypes with digital testing equipment, interfaces are used in the design so that chip I/O is performed in voltage form. In Fig.8, high voltages (5V) correspond to +2μA, while low voltages (0V) correspond to -2μA. Boundary cells are set to a low state value. Note that convergency time is only 3.5μs, with 16 cells in a row. Finally, Montecarlo simulations showed 90% and 100% insensibility to process and geometries variations (out of 30 trials) for prototypes P1 and P2 respectively.

Conclusions

A general procedure for the design of continuous and discrete time CNNs using current-mode techniques has been presented. The circuits obtained with this methodology are advantageous in terms of area, speed and power over previous implementation techniques. Several prototypes have been designed, the results been highly insensitive to process and design-parameter variations.

References

[Chua88a] L.O. Chua and L. Yang: "Cellular Neural Networks: Theory". *IEEE Trans. Circuits and Systems*, Vol. CAS-35, pp 1257-1272, 1988.

[Chua88b] L.O. Chua and L. Yang: "Cellular Neural Networks: Applications". *IEEE Trans. Circuits and Systems*, Vol. CAS-35, pp 1273-1290, 1988.

[Cruz91] J.M. Cruz and L.O. Chua: "A CNN Chip for Connected Component Detection". *IEEE Trans. Circuits and Systems*, Vol. CAS-38, pp 812-817, 1991.

[Halo92] K. Halonen et al: "VLSI Implementation of a Reconfigurable Cellular Neural Network Containing Local Logic". *Int. J. Circuit Theory Applications*, 1992 (to appear)

[Harr92] H. Harrer et al.: "An Analog Implementation of Discrete-Time Cellular Neural Networks". *IEEE Trans. Neural Networks*, Vol. 3, pp 466-476, 1992.

[Huer92] J.L. Huertas et al.: "Analog VLSI Implementation of Cellular Neural Networks". *Second Int. Workshop on CNNs and their Applications*, Munich, 1992.

[Sayl91] A.H. Sayles et al.: "An Optoelectronic CMOS Memory Circuit for Parallel Detection and Storage of Optical Data". *IEEE J. Solid State Circuits*, Vol. 26, pp 1110-1115, 1991.

[Toum90] C. Toumazou et al.: *"Analogue IC Design: the Current Mode Approach"*. Peter Peregrinus Ltd., London 1990.

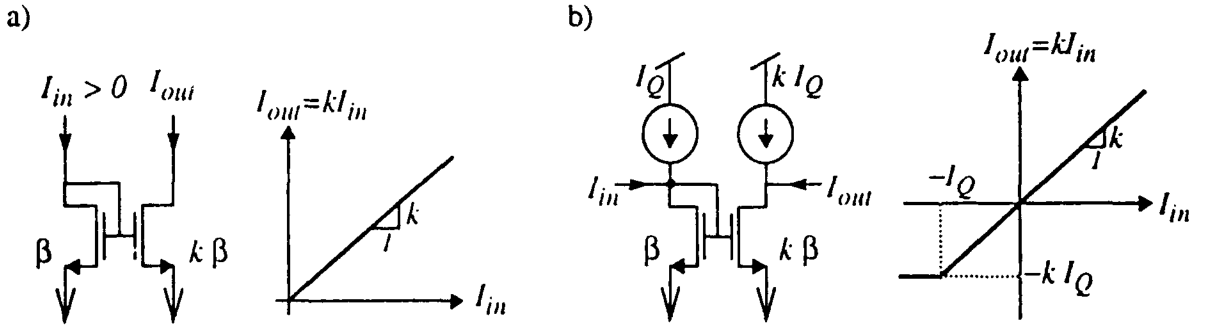


Figure 1: a) MOS current mirror; b) MOS current mirror with bias shifting

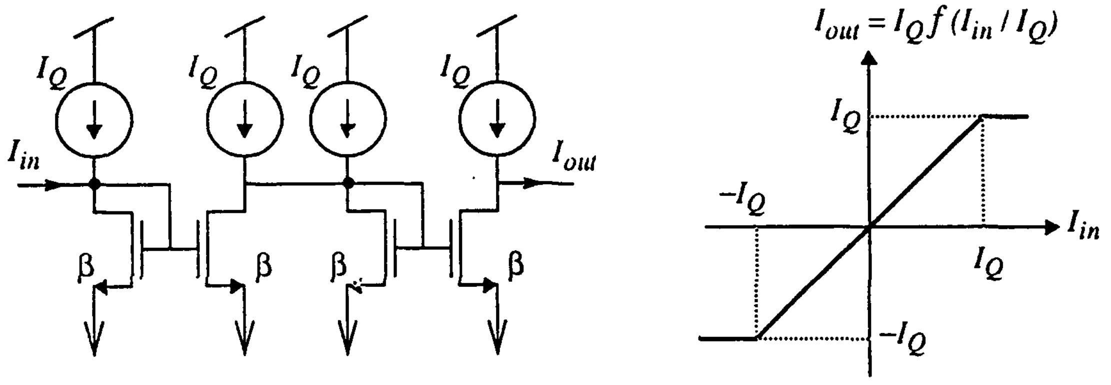


Figure 2: Implementation of the nonlinear operator $f(\bullet)$

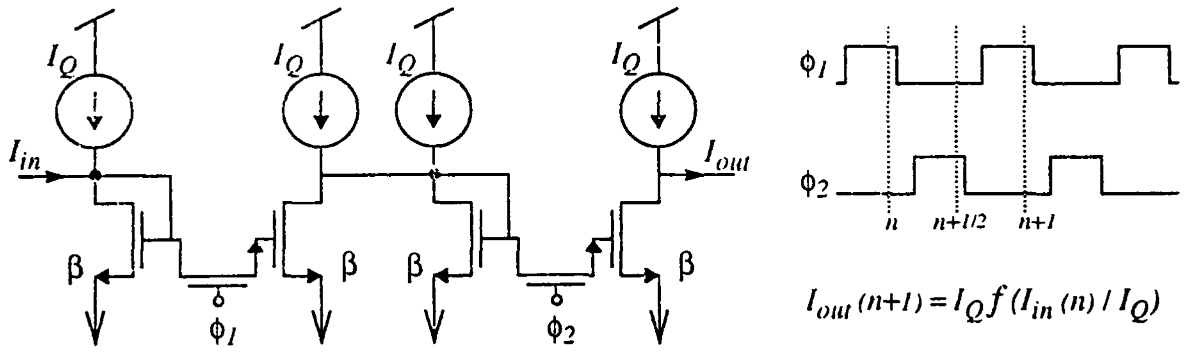
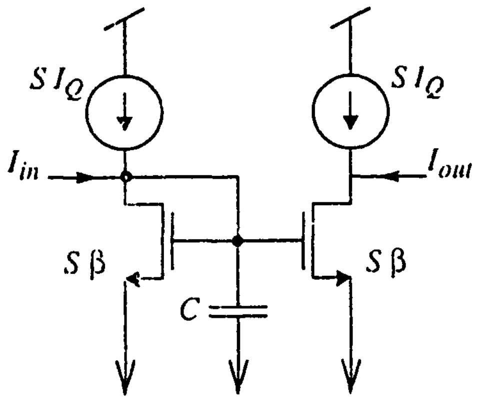


Figure 3: Implementation of the DT dynamic operator: a one clock-cycle delayer



$$\frac{C}{g_m}\frac{dI_{out}}{dI_{in}} = -I_{out} + I_{in}$$

$$I_{in}, I_{out} > -S I_Q$$

Figure 4: CT lossy integrator

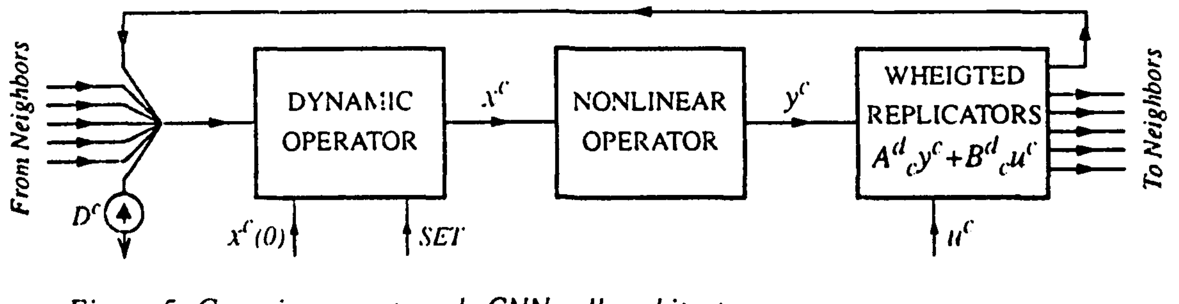
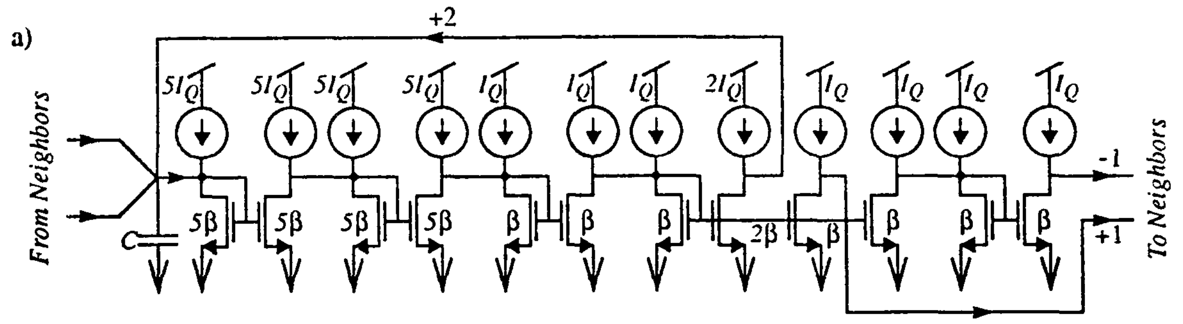


Figure 5: Generic current-mode CNN cell architecture



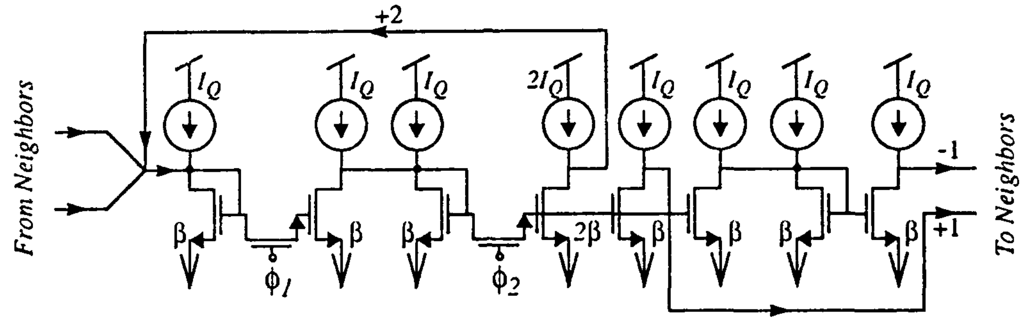


Figure 6: Schematic of a CNN cell for CCD: a) Continuous Time; b) Discrete Time

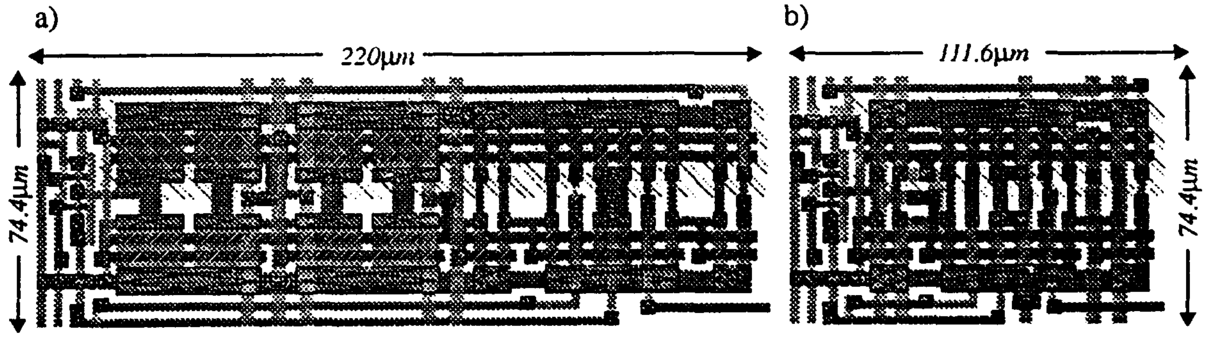


Figure 7: Cell layout for two CCD prototypes: a) prototype P1; b) prototype P2

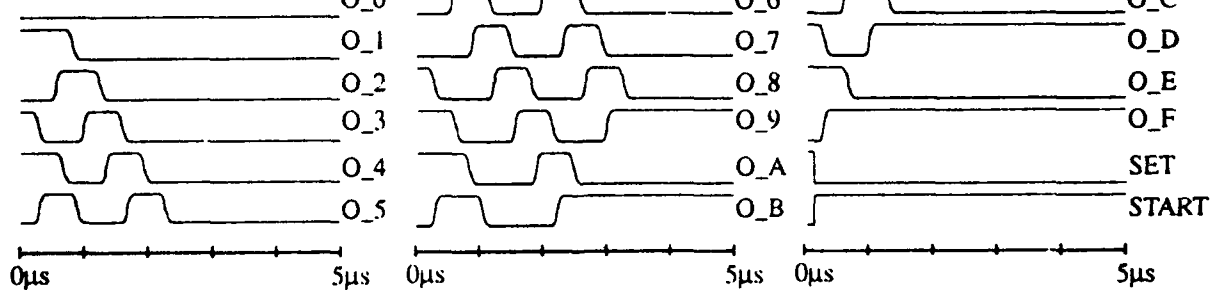


Figure 8: Electrical simulation results from extracted netlist of prototype P1

Optically Realized Feedback Cellular Neural Networks

Krzysztof Ślot
Institute of Electronics, Technical University of Łódź,
Stefanowskiego Str.18/22, 90-924 Łódź, Poland

Abstract

This paper is concerned with an optical implementation of Cellular Neural Networks, defined by a template with non-zero feedback operator. Two design rules which consider a transient duration of optically realized CNNs are formulated. A possibility of an increase in a resolution of images, which are to be processed in the optical system, is also discussed. The rules formulated in the paper are applied into a CNN design procedure – an optical implementations of two distinct Hole Filling CNNs and a Shadow Detecting CNN are analyzed.

1 Introduction

The Cellular Neural Network paradigm (abbreviated henceforth CNN) [1] has attracted a lot of interest because it offers a natural way of solving many of image processing problems and a feasibility of simple CNN circuits (as e.g. [2]) VLSI implementation. However, problems with implementing highly interconnected CNNs composed of large number of cells are severe. An alternative mean for a physical CNN realization, which allows for overcoming these difficulties, is the optical system proposed in [3] and [4]. Specific properties of the optical system should be considered in deriving CNN applications which are to be realized optically, since then we can exploit the system advantages and minimize its drawbacks.
The main subject of this paper is a determination of ways which allow for increasing a processing speed of optically implemented *feedback CNNs*. A CNN is referred to as a *feedback* one if there is at least a single non-zero element in the feedback operator (A) of the circuit template (see [1]). Another issue which is discussed in the paper is a possible increase in a resolution of images which are to be processed.
The main results of the paper are following:

- Design rules which can be incorporated into a CNN design procedure and which concern a convergence speed of optically realized feedback CNNs are presented. Conditions which allow for an increase in a resolution of images which are to be processed in the system are formulated.

0-7803-0875-1/92 3$.00 ©1992IEEE

The paper has the following structure. Basic processing properties of the optical system are outlined in Section 2. The design rules, which concern a processing speed of optically realized CNNs, are formulated in Section 3. Section 4 considers a possibility of an increase in a resolution of images which are to be processed in the system. The results of presented considerations are applied to analyze an optical implementation of some of existing feedback CNN applications in Section 5.

2 Processing properties of the optical system

The optical system implements physically a CNN model which combines the discrete-time version of the *cell state equation* [1] (where a time step is chosen to be: $\Delta t = \frac{1}{RC}$, as proposed in [5]):

$$Vx_{ij}(n+1) = \sum_{k=-r}^{r} \sum_{l=-r}^{r} A_{kl} \cdot Vy_{k+i,l+j}(n) + \sum_{k=-r}^{r} \sum_{l=-r}^{r} B_{kl} \cdot Vu_{k+i,l+j} + I \qquad (1)$$

and a sigmoidal output transforming function, which approximates the *cell output equation* [1]. The symbols Vx_{ij}, Vy_{ij}, Vu_{ij}, A_{kl}, B_{kl}, I have the meaning defined in [1].
The most attractive properties of the optical system are: a realizability of CNNs composed of a large number of cells (which means a possibility of high resolution image processing) and an ease of large neighborhood-size templates realization. However, it should be noticed, that due to a special coding scheme ([3],[4]), a maximum resolution of images which can be processed in the system is generally four times lower than the available resolution of the system input device (Liquid Crystal Light Valves array).
A presence of feedback allows for exploiting a global information by every circuit cell, despite a local interconnection pattern. In the case of discrete-time CNNs this interaction at large distances occurs indirectly, through cell outputs, in consecutive iterations. Realization of every iteration requires closing a feedback loop, i.e. transferring results obtained in some earlier step from the system output back to its input. This is highly time-consuming operation, which severely slows down a system processing speed (computations are performed at the speed of light), and can be regarded as the main drawback of the optical system. Therefore the following conclusion concerning processing properties of the optical system can be formulated:

- Applications which require a large neighborhood-size and a small number of iterations to complete a network transient are well suited for the optical CNN realization.

3 A processing speed of optically realized feedback CNNs

The optical system, despite its drawbacks, can appear the only physical mean of some CNN circuits realization. Therefore it is necessary to investigate possible ways of improving system properties, in particular, ways of increasing its processing speed. Some CNN design methods for feedforward-only CNNs has been presented in [6]. This paper focuses on a feedback CNN template design.
Let us consider the following process:

a cell c_{ij} output follows a transition from '-1' to '+1', which occurs in the output of any of its neighboring cells, unless some condition related to inputs analyzed by a cell c_{ij} is satisfied, -
and let us refer it to as a *high output propagation.* A *low output propagation* is understood analogously. These phenomena are present in many CNN applications introduced to date ([7], [8], [9]). If a CNN which is to be implemented optically involves a propagation phenomenon, it should be designed in a way which ensures completing this process in a minimum possible number of iterations. Therefore, some means which allow for obtaining this goal should be included into a CNN design procedure.
A simple analysis of the propagation phenomena allows to determine some template properties which allow for an increase in a processing speed of optically realized CNNs. First, observe that if a given cell is allowed to change its output, this change should have the maximum possible magnitude in response for any change occurring in outputs of its neighborhood. This means that a change of an absolute value: $\Delta Vy = Vy_{max} - Vy_{min}$ (i.e. $\Delta Vy = 2$ for $Vy_{min} = -1$ and $Vy_{max} = +1$) should occur in a single iteration. This leads to the following conclusion, which can be considered as an auxiliary CNN design rule:

Rule 1 *Suppose, that we have a processing problem which involves a propagation of high or low output and which is to be solved in an optically realized CNN.*
Template element values should be chosen to ensure a binary output from any cell in response to any combination of cell controlling signal values which can appear in this network, i.e.:

$$|\sum_{kl} A_{kl} \cdot Vy_{k+i,l+j}(n) + \sum_{kl} B_{kl} \cdot Vu_{k+i,l+j} + I| \geq 1 \qquad (2)$$

Note, that if this condition is satisfied, all cell outputs are binary during a processing, so a CNN behaves as if it is composed of elements with a step-like output nonlinearity
Observe further, that if we increase a cell neighborhood-size then a cell can respond for a change occurring in more distant location. This reasoning leads to a conclusion:

Rule 2 *Maximum neighborhood size, which allows for realizing given processing problem, should be considered in deriving CNN templates to ensure minimizing of a transient duration in optically realized feedback CNNs.*

4 A resolution problems in optically realized feedback CNNs

Let us consider a coding scheme for signals which are to be processed in the optical system, presented in [3],[4]. An input information loaded into the system is stored according to this scheme in an input Liquid Crystal Light Valve array (abbreviated LCLV). Half of all of the available LCLV array elements are used for storing values which are to provide a required shift in a signal dynamic range. This shift includes a CNN bias (I), but primarily it is to allow for a discrimination between negative and positive results of optical computations. A result of the optically computed cell state (1) is represented by a light vector Vx^l_{ij}, which has the magnitude given by:

$$Vx^l_{ij}(n+1) = k \cdot (\sum_{kl} A_{kl}(Vy_{k+i,l+j}(n) + 1) + \sum_{kl} B_{kl}(Vu_{k+i,l+j} + 1) + d + I) \qquad (3)$$

where k is some positive constant, d represents an additional shift, and terms '+1' are introduced because transmissivities of light valves, which are proportional to CNN signals, are non-negative.
Let us assume that a CNN bias (I), which can be considered as a shift in a dynamic range of the state equation (1), is realized in an electronic feedback path of the system (by means of a level shifter) rather then in an optical signal path. It can be shown that errors, caused by inability of discrimination between negative and positive magnitudes of vectors Vx^l_{ij}, can be neglected in a signal processing if the following relation holds:

$$|Vx^l_{-1}| > |Vx^l_{min}| \tag{4}$$

where Vx^l_{-1} is a magnitude of a light vector which corresponds to the value '-1' of the state equation (1); Vx^l_{min} is a magnitude of the light vector which corresponds to a minimum value of the state equation.
Note that, if there is no entries which code a bias and shift in the LCLV array, a zero-level of optically computed cell's state is in fact shifted by a value of: $k \cdot (\sum A_{kl} + \sum B_{kl} - I)$. Thus, magnitudes of Vx^l_{-1} and Vx^l_{min} can be evaluated using the formulas:

$$\begin{array}{c} Vx^l_{-1} = k \cdot (\sum A_{kl} + \sum B_{kl} - I - 1) \\ Vx^l_{min} = k \cdot (-2\sum |A_{kl}| - 2\sum |B_{kl}| - I) \quad \text{for } A_{kl}, B_{kl} < 0 \end{array} \tag{5}$$

The following conclusion can be formulated:

- If the light vectors given in equation (5) satisfy the relation (4) then it is possible to realize a bias in an electronic part of the system, and consequently, to double a resolution of images which are to be processed in the system.

Observe, that if the relation $\sum A_{kl} + \sum B_{kl} = I$ holds, and the condition (4) is satisfied, then no shift is required in the system.

5 A design of optically realized CNNs

Templates for two of already existing CNN applications are analyzed in this section in the way which takes into account a convenience of their optical implementation. The CNNs considered are: the Hole Filler, presented in [7] and [10], and the Shadow Detector, presented in [7].

5.1 A modification of the Hole Filler template

An analysis of the Hole Filler operation shows, that only the first of the formulated rules can be applied into a template design procedure. One can notice, that comparing two proposed hole filler templates ([7] and [10]), only the first one (proposed by Matsumoto *et all.*) complies to the first rule, so it is expected to ensure the faster convergence in the optical system. Results of computer simulations which verify this hypothesis are shown in Figure 1. An image which is to be processed is given in Figure 1a, a processing result is shown in Figure 1c. This is obtained in 50 iterations when the first of considered templates is used, while it takes 60 iterations to reach this result in the latter case. Figure 1b presents the CNN output after 50 iterations in the case when the template given [10]

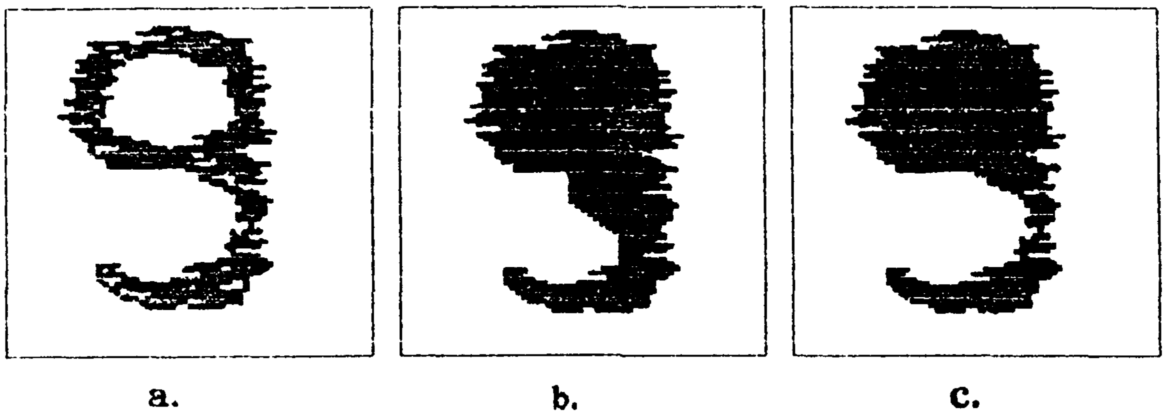


Figure 1: A comparision of the hole filling operation

is used. Initial states of all boundary network cells are set to '-1' and of the remaining ones are set to '+1'.
Observe, that template elements of both hole filler templates satisfy the condition (4): $V^l_{min} = -1$, $V^l_{-1} = 10$ in the former case, and $V^l_{min} = 0$, $V^l_{-1} = 10.5$ in the latter case. Therefore, if we realize an appropriate signal range shift in an electronic path of the system, a resolution of images which are to be processed can be doubled.

5.2 A modification of the Shadow Detector template

The second example shows the application of both rules in order to derive an alternative template of a shadow detector. Let us assume that we increase a neighborhood size of a template (i.e. $r = 2$ instead of $r = 1$ as it was given in [7]). Applying the first of the formulated rules into a design strategy proposed in [10], the following element values can be obtained:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \end{bmatrix} \quad I = 2$$

A processing example given in Figure 2 compares an operation of a CNN defined by this template and an operation of the original Shadow Detecting CNN. An image to be processed is presented in Figure 2a, the processing result, which is obtained in the former case after 14 and in the latter case after 28 iterations, is given in Figure 2c, while the output of the original Shadow Detector after 14 iterations is shown in Figure 2b. Appropriate templates for neighborhood size larger than $r = 2$ can be easily derived. A correct processing can be obtained if it is assumed that after every iteration right boundary cell states (outputs) are set to '-1' regardless of computation results.
Note, that a resolution of images to be processed in the case of proposed template equals a resolution of a LCLV device if an appropriate shift is realized in the feedback path of the system.

6 Conclusions

Possible methods of improving processing properties of optically realized CNNs were presented in the paper. An application of two design rules, which has been formulated, can

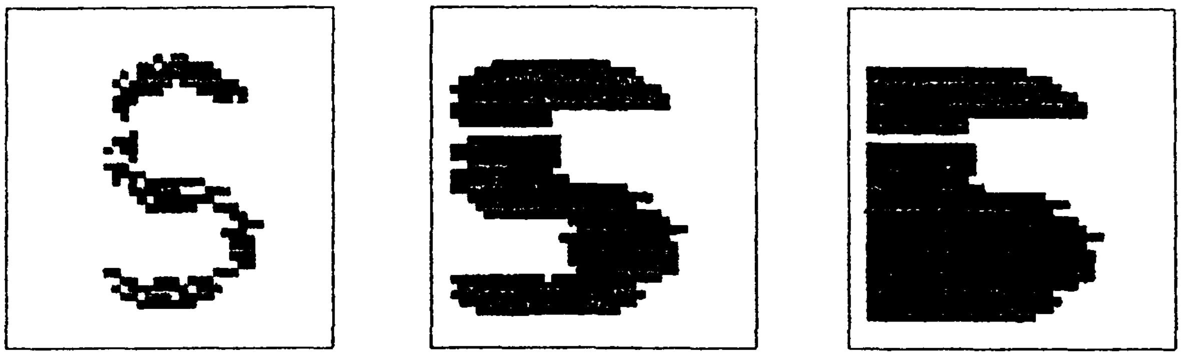


Figure 2: An operation of a shadow detection

result in an increase in the processing speed of some optically realized feedback CNN applications. A realization of a signal dynamic range shift in an electronic feedback path of the system can allow for an increase in a resolution of images which are subject to optical processing.

Acknowledgements

The author wish to thank to prof.T.Kacprzak for his help in preparation of this paper.

References

[1] L.O.Chua, L.Yang *Cellular Neural Networks: Theory and Appliccations* IEEE Trans. on Circuits and Systems, vol.35, 1257-1290, 1988.

[2] J.M.Cruz, L.O.Chua *A CNN Chip for Connected Component Detection* IEEE Trans. on CaS, vol.38, No.7, pp 812-817, July 1991.

[3] N.Frühauf, E.Lüder *Realizations of CNNs by Optical Parllel Processing with Spatial Light Valves* CNNA-1990 IEEE Proc. pp281-291, 1990.

[4] N.Frühauf, E.Lüder, M.Gaida, G.Bader *An Optical Implementation of Space Inavriant Cellular Neural Networks* ECCTD-1991 IEEE Proc. pp42-51, 1991.

[5] H.Harrer, J.A.Nossek *Time Discrete Cellular Neural Networks: Architecture, Applications and Realization* Rep. No.:TUM-LNS-TR-90-12, Technical University of Munich, Nov 1990

[6] K.Ślot, T Roska, L.O.Chua *Optically Realized Feedforward-Only Cellular Neural Networks* AEÜ vol.46, No.3. pp 158-168, May 1992, Hirzel Verlag, Stuttgart.

[7] T.Matsumoto, T.Yokohama, H.Suzuki, R.Furukawa *Several Image Processing Examples by CNN* CNNA-1990 IEEE Proc. 100-111, 1990.

[8] L.O.Chua, B.Shi *Multiple Layer Cellular Neural Networks - A Tutorial* Memo UCB/ERLM90-113, Univ. of Cal. at Berkeley, Dec. 1990.

[9] K.Ślot *Cellular Neural Network Design for Solving Specific Image-Processing Problems* to appear in Special Issue of CTA on CNNs, vol.20, 1992

[10] L.O.Chua, P.Thiran *An analythic Method for Designing Simple Cellular Neural Networks* IEEE Trans. on CaS, vol.38, No.11, November 1991.

THE CNN UNIVERSAL MACHINE

Part 2: Programmability and Applications

Tamás ROSKA+ and Leon.O.CHUA

The Electronics Research Laboratory and Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley CA 94720 and the +Dual and Neural Computing Systems Laboratory in the Computer and Automation Institute of the Hungarian Academy of Sciences, Budapest, Kende-u.13/17, H-1111

Abstract

The programmability (as a stored program) of the CNN Universal Machine is discussed first. It is shown why and in which sense this machine is universal. A new type of algorithm, the analogic one, is introduced. The application potential is reviewed and the biological relevance is analyzed. It turns out that the architecture is optimal not only for silicon implementations, however, many biological information processing organs have the same structure.

1 INTRODUCTION

Since the invention of the Cellular Neural Network [1] several extensions [2-7, etc.] formed the CNN (cellular nonlinear network) paradigm. CNN is now an established field. In our companion paper in this volume (Part 1) we gave a taxonomy and introduced the architecture of the CNN Universal Machine.

In this paper we show the key part responsible for implementing the analog stored program concept (Section 2). In Section 3 we discuss the universality of this machine and the characteristics of the analogic (dual) CNN algorithms. In Section 4 the key application areas are reviewed. Section 5 contains some examples and motivations concerning the biological relevance.

2 THE ANALOG STORED PROGRAM IN THE CNN UNIVERSAL MACHINE

The key issue in inventing the digital computer was to represent the program as data (J.von Neumann). A condition for efficient implementation is that the time for changing the instructions is normally smaller or comparable to the instruction execution time.

Considering the standard fully connected analog neural networks, it is obvious that the latter condition is unimplementable. Partly because the storage space of a single program "instruction" is prohibitively large, partly because the reprogrammability takes more time than the execution time itself. This is a point where the local connectivity comes into the picture. Namely, if we consider a CNN cloning template as an analog instruction [2d], then we have only one or two dozen analog values for an instruction to be stored even if we have thousands of processors. On the contrary, in case of a fully connected analog synchronous neural network (e.g. the Intel 80170 [9]) we have about 10 000 analog values for 64 processors and their reprogrammability takes considerable time.

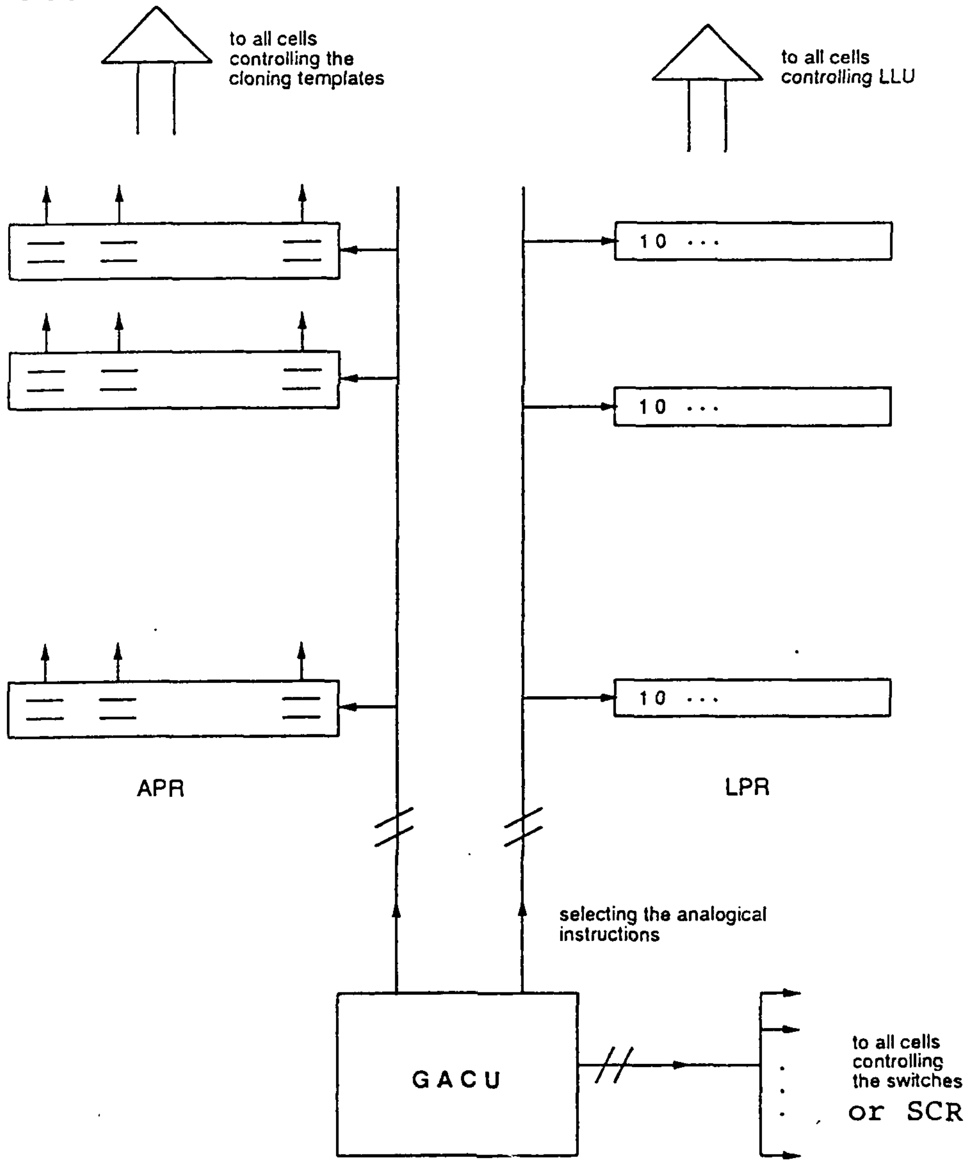

to all cells controlling the cloning templates
to all cells controlling LLU
1 0 ...
1 0 ...
1 0 ...
APR
LPR
selecting the analogical instructions
G A C U
to all cells controlling the switches
or SCR

Figure 1 The GAPU

The analog instructions in the CNN Universal Machine are stored in the analog program register (APR) of the global analogic program unit (GAPU). The structure of the GAPU is shown in Figure 1.

The analog memory elements in a single line of the APR is equal to the nonzero cloning template parameters. The selection of the appropriate template is made by the global analogic control unit (GACU) which is a logic unit. The logic program register contains the local logic instructions. The switch configurations of the cells, including the selection of the appropriate local analog output unit (LAOU, defined in Part 1) can be coded and stored in the SCR register (switch configuration register). The machine code of the analogic CNN program is stored in the GACU.

The high level description of the analogic algorithm is translated by a compiler into this machine code.

3 UNIVERSALITY AND THE ANALOGIC ALGORITHMS

We have shown [2c] that the game of life can be implemented by a CNN. Since the game of life algorithm is equivalent to a Turing machine, therefore the CNN is universal in a logic sense. As an analog operator, we have shown that any nonlinear operator with fading memory can be realized by delay-type neural networks [8a]. Here, the open question is: how many elements do we need?

The flexibility of the CNN is not only theoretical. Indeed, using our CNN Workstation it is easy to write and run analogic CNN programs. As an example, Figure 2 shows such an algorithm in a block diagram form. The details of the algorithm can be found in [2c]. It has been used for detecting texture errors in textiles monitored through a camera. Here only the structure of the algorithm is important. In this algorithm we have parallel branches as well. A key point in the implementation of the analogic (dual) CNN algorithms that all intermediate results which will be used later are stored locally as analog values. Therefore, in the cell of the CNN Universal Chip a local analog output memory is used to store as many analog values as the maximum number of parallel branches. The local storage provides extremely high speed. We need to output the results after the whole analogic algorithm is completed, thus providing a high output throughput.

Now, we have analogic algorithms, instructions, subroutines, programs, compilers. This provides to build up a CNN software library. The CNN universal Machine is the workhorse for running the analogic CNN programs.

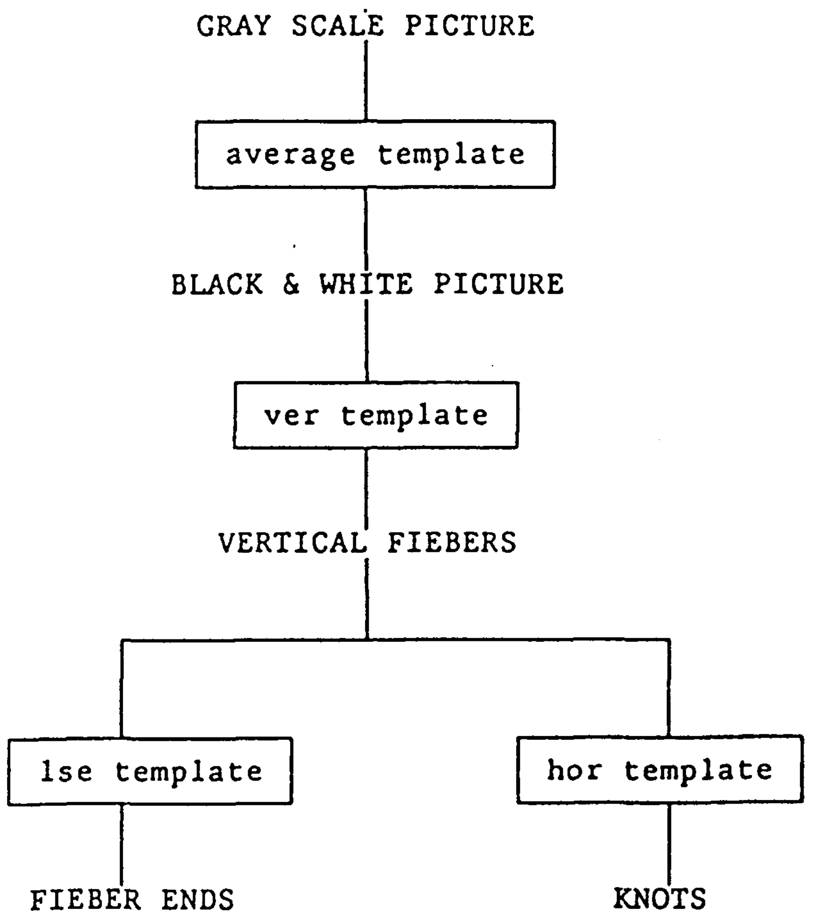


Figure 2 The flow diagram of an analogic (dual) CNN algorithm

4 KEY APPLICATION AREAS

In the CNN literature (see [7a-7e] and the references of Part1) we see how many application areas have been discovered. Without completeness let us list some major fields:

- Image processing including gray scale image inputs
    - feature extraction
    - motion detection and estimation
    - path tracking
    - collision avoidance
    - halftoning including motion
    - object counting and size estimation

- Analyzing 3D surfaces
    - detection minima and maxima
    - detection areas where the gradients exceed a given limit
- Solving partial differential equations
- Reducing non-visual problems to geometric maps
    - thermographic images
    - somatosensory maps (tactile sensing)
    - antenna array images
    - different medical maps and images
- Modelling biological vision and other sensory-motor organs

System integration

Since presently the analog VLSI CNN chips are mainly small scale experiments, and not considering the software tests of key effects for different applications, the few emerging real-life application projects known to us are using digital CNN hardware accelerators.

Detecting manufacturing errors in a printed circuit board production line, multifont character recognition, and optical tracking of prescribed curves in robot motion were the three projects we have started. The details are published elsewhere [7a-e]. However, we have found some common problems. Namely,

- the throughput problem of the input interface (the ideal solution is the direct optical input on the chip, or the direct camera interface for the digital CNN hardware accelerators),
- the changing illumination circumstances and colour effects at the actual site (different materials in the printed circuit manufacturing, the different qualities of papers and prints in case of character recognition, and the changing environment in case of the robots),
- the throughput problem at the output, especially in detection type tasks,
- the optimal implementation of the analogic (dual) CNN algorithms as to the area (memory) and time complexity is concerned.

4 BIOLOGICAL RELEVANCE

In many living neural organizations the neurons are placed regularly in laminae packed upon each other, and the neurons are locally connected within a receptive field.

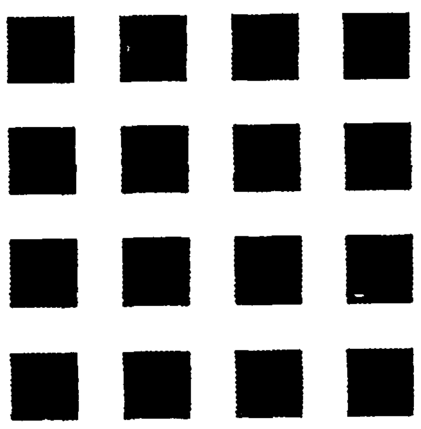
The functionality of the organ (e.g. retina, LGN or visual cortical area) is determined by the neuroanatomical receptive field organization and by the synapse types. The CNN model, invented for artificial electronic information processing, has the same structural and functional characteristics. Until recently, both areas were developing independently. It was our fortune that in a nice and challenging collaboration [25] it turned out that almost all relevant notions in both fields can be matched. Thus it is our hope, and indeed the reality, that some known neural structures can be translated into CNN models which have the one-to-one silicon chip implementations. Moreover, apparently, the CNN model could play a role in discovering hidden living functionalities as being a unifying programmable model for many biological phenomena [26]. The receptive field organization is represented by the cloning template of the CNN.

A few years ago there were a couple of electronic circuit models discovered and implemented on silicon chips to mimic specific neural phenomena. A famous example is the so called silicon retina of Carver Mead's group at Caltech in Pasadena [13]. Indeed it is a very special case of a CNN model, a resistive grid characterized by a simple cloning template.

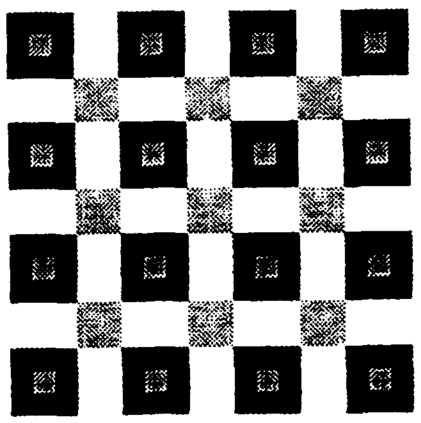
Neuromorphic computing is nowadays a new challenge [27]: how to translate one-to-one a living neural structure into a programmable analog computing chip. The CNN Universal Machine and Chip is providing a solution. We have found also that the famous triad synapse model [14] can be represented by a simple CNN template, playing important role in motion related actions as well.

Below we show some characteristic examples [25b]. The Herring grid illusion without the central patches (taking into account the amacrine cell effects) and the arrow head illusion with its CNN template.

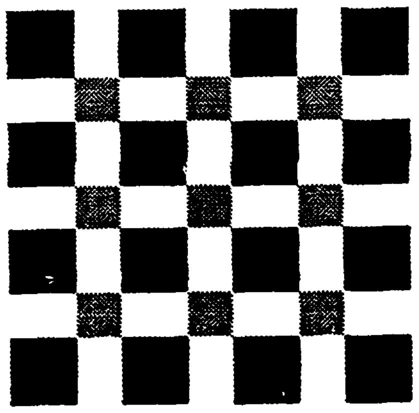
Another important fascinating area is the so called multi screen theater [18]. The fact is that from the same scene, preprocessed by the eye, several different retinotopic maps are generated [17,19,28]. Later, these maps are combined to discover more complex events. This phenomenon can be directly modelled and computed by the CNN universal chip.



(a)



(b)



(c)

Figure 3 The Herring grid illusion.

The input we look at (a), the result using a resistive grid (b), and the result when modelling the amacrine cell layer by delay-type templates (c). Indeed, we do not see the patches in the middle of the black squares.

$$A = [1.3] \quad B = \begin{bmatrix} -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\ -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\ -0.1 & -0.1 & 1.3 & -0.1 & -0.1 \\ -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\ -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \end{bmatrix} \quad I = 0$$

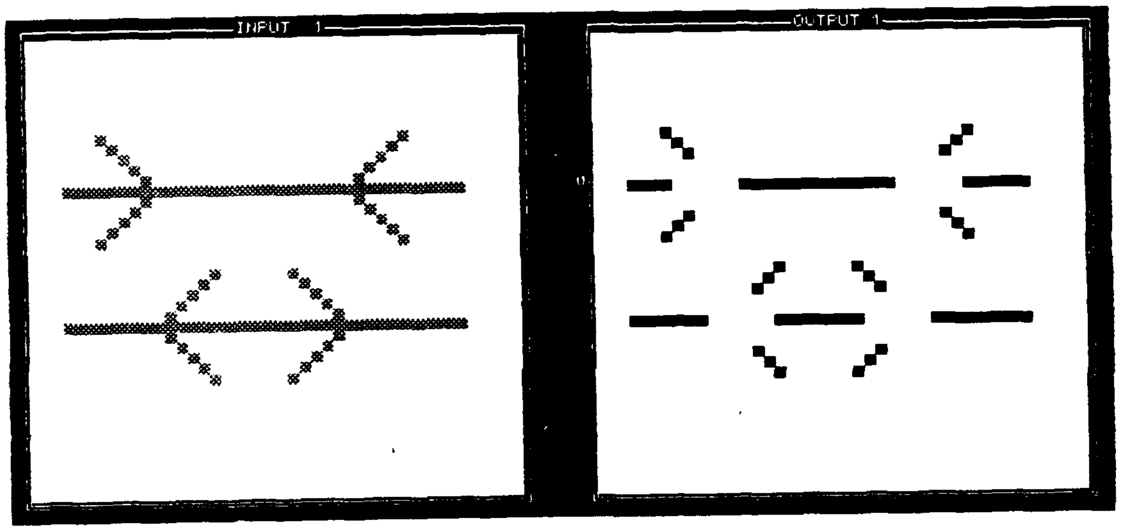


Figure 4 The arrowhead illusion and the CNN template

Although the distances between the arrowheads are the same in both cases we see defferent distances.

ACKNOWLEDGEMENTS

This work has been supported by the Office of Naval Research under Grant N00014-89-J-1402, by the National Science Foundation under grant MIP 86-14000, and by the National Science Foundation under Grant INT 90-01336 in cooperation with the Hungarian Academy of Sciences.

REFERENCES

[1a] L.O.Chua and L.Yang, "Cellular neural networks: Theory", IEEE Transactions on Circuits and Systems, Vol.35, pp.1257-1272, 1988

[1b] L.O.Chua and L.Yang, "Cellular neural networks: Applications", ibid., pp.1273-1290.

[2a] T.Roska and L.O.Chua, "Cellular neural networks with nonlinear and delay-type template elements", Proc. IEEE CNNA-90, pp.12-25, 1990

[2b] T.Roska and L.O.Chua, "Cellular neural networks with nonlinear and delay-type template elements and non-uniform grids", Int.J.Circuit Theory and Applications, to appear, Vol.20, 1992

[2c] L.O.Chua, T.Roska P.L.Venetianer and Á.Zarándy, "Some Novel Capabilities of CNN: Game of Life and Examples of Multipath Algorithms", Report DNS-3-1992 Dual and Neural Computing Systems Laboratory, Computer and Automation Institute of the Hungarian Academy of Sciences, Budapest, 1992, paper in this Proceedings

[2d] T.Roska and L.O.Chua, "CNN: Cellular analog programmable multidimensional processing array with distributed logic and memory", submitted for publication, Report DNS-2-1992 (ibid)

[3] J.A.Nossek, G.Seiler, T.Roska and L.O.Chua, "Cellular neural networks: Theory and circuit design", Report No.:TUM-LNS-TR-90-7, Inst. Network Theory and Circuit Design, T.U.Munich, Munich December, 1990 and Int.J. Circuit Theory and Applications, to appear, Vol.20, 1992

[4] H.Harrer and J.A.Nossek, "Time discrete cellular neural networks: architecture, applications and realization", Report No. TUM-LNS-TR-90-12, Technical University of Munich, November 1990, revised version in Int. J. Circuit Theory and Applications, to appear, Vol.20, 1992

[5] J.Henseler and P.J.Braspenning, "Membrain: a cellular neural network model based on a vibrating membrane", Int.J. Circuit Theory and Applications, to appear, Vol.20, 1992

[6] A.Radványi, K.Halonen and T.Roska, "The CNNL simulator and some time-varying CNN templates", Report DNS-9-1991, Dual and Neural Computing Systems Laboratory, Comp. Aut. Inst., Hungarian Academy of Sciences, Budapest, 1992

[7a] Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA-90, Budapest, 1990, IEEE Cat.No. 90TH0312-9

[7b] Special Session on Cellular Neural Networks, Proc. ECCTD-91, European Conference on Circuit Theory and Design, Copenhagen, September 1991

[7c] Special issue on Cellular Neural Networks, Int.J.Circuit Theory and Applications, Vol.20, 1992, in print

[7d] [8a] Proceedings of the 2nd IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA-92, Munich, October, 1992 (this Proceedings)

[7e] Special Issue on Cellular Neural Networks, IEEE Transactions on Circuits and Systems, to be published in March, 1993

[8a] T.Roska, "Analog events and a dual computing structure using analog and digital circuits and operators", pp.225-238 in Discrete Event Systems: Models and Applications (eds. P.Varaiya and A.B.Kurzhanski), Springer-Verlag, Berlin, 1988.

[8b] T.Roska, "Dual computing structures containing analog cellular neural networks and digital decision units", Proc. IFIP Workshop on Silicon Architectures for Neural Nets, Nice, 1990, pp.233-244, North Holland, Amsterdam, 1991

[9] Intel 80170 XA Electrically Trainable Analog Neural Network, Advance Information, Intel Corp., 1990

[10] J.M.Cruz and L.O.Chua, "A CNN chip for connected component detection", IEEE Trans. Circuits and Systems, Vol.38, pp.812-817, 1991

[11] V.I.Krinsky, V.N.Biktashev and I.R.Efimov, "Autowave principles for parallel image processing", Physica D, Vol.49, pp. 247-253, 1991

[12] C.B.Price, P.Wambacq and A.Osterlinck, "Image enhancement and analysis with reaction-diffusion paradigm", IEE Proceedings, Vol.137, Pt.I, pp.136-145, 1990

[13] M.A.Mahowald and C.Mead, "The silicon retina", Scientific American, Vol.264, No.5, pp. 76-82, May 1991

[14] J.Hámori, T.Pasik, P.Pasik and J.Szentágothai, "Triadic synaptic arrangements and their possible significance in the lateral geniculate nucleus of the monkey", Brain Research, Vol.80, pp.379-393, 1974

[15] E.Lábos, "Theoretical considerations of local neuron circuits and their triadic synaptic arrangements (TSA) in subcortical nuclei", J.Neuroscience Research, Vol.3, pp. 1-10, 1977

[16] W.Heiligenberg, "The neural basis of behavior: a neuroethological view", Ann. Rev. Neurosci., Vol.14, pp.247-267, 1991

[17] J.Allman, F.Miezin and E. McGuiness, "Stimulus specific responses beyond the classical receptive field" neurophysiological mechanisms for local-global comparisons in visual neurons", Ann. Rev. Neurosci., Vol.8, pp.407-430, 1985

[18] G. Montgomery, "The mind's eye", Discover, May 1991

[19] D.C.Van Essen, C.H.Anderson and D.J.Felleman, "Information processing in the primate visual system: an integrated systems perspective", Science, Vol.255, pp.419-423, January 24,1992

[20] J.L.Bradshaw and N.C.Nettleton, Human cerebral assymmetry (Chapter Nine), Prentice Hall, Englewood Cliffs, 1983

[21] E.R.Kandel and J.H.Schwarz, Principles of neurai science, Elsevier, Amsterdam, 1985

[22] J.E.Dowling, The Retina: An approachable part of the brain, Harward University Press, Cambridge, Massachusetts, 1987

[23] F.S.Werblin, The control of sensitivity in the retina, Scientific American, Vol.228, pp.71-79, 1973

[24] F.Wörgötter and C.Koch, "A detailed model of the visual pathway in the cat: Comparison of afferent excitory and intracortical inhibitory connection schemes for orientation slectivity", The J. Neuroscience, Vol. 11, pp. 1959-1979, 1991

[25a] T.Roska, K.Lotz, J.Hámori, E.Lábos and J.Takács, et. al., "The CNN model in the visual pathway - Part I: The CNN-Retina and some direction- and length-selective mechanisms", Report DNS-10-1991, Dual and Neural Computing Systems Res.Laboratory, Comp.Aut.Inst., Hungarian Academy of Sciences, Budapest, 1991

[25b] T.Roska, J.Hámori, E.Lábos, K.Lotz, L.Orzó, J.Takács, P.Venetianer, Z.Vidnyánszky, Á.Zarándy, "The CNN model in the visual pathway - Part II:The amacrine cell in the modified retina model, simple LGN effects and motion related illusions", Report DNS-9-1992, Dual and Neural Computing System Res. Laboratory, Comp.Aut.Inst., Hungarian Academy of Sciences, Budapest, 1992, under publication

[26] W.Heiligenberg and T.Roska , "On biological sensory information processing principles relevant to dual computing CNNs", Report DNS-4-1992, Dual and Neural Computing Systems Res.Lab., Comp. Aut. Inst., Hungarian Academy of Sciences, Budapest, 1992, under publication

[27] Workshop on Unifying Models of Biological Information Processing versus Neuromorphic CNN Models, Berkeley, August 3, 1992

[28] Special Issue on Mind and Brain, Scientific American, September, 1992

JAPANESE KANJI CHARACTER RECOGNITION USING CELLULAR NEURAL NETWORKS AND MODIFIED SELF-ORGANIZING FEATURE MAP

Kenji NAKAYAMA Yasuhide CHIGAWA

Dept. of Electrical and Computer Eng., Faculty of Tech., Kanazawa Univ.
2-40-20, Kodatsuno, Kanazawa, 920 JAPAN
Tel +81-762-61-2101 Ext.372; Fax +81-762-61-2509
E-mail nakayama@haspnn1.ec.t.kanazawa-u.ac.jp

ABSTRACT The cellular neural networks for extracting line segment features are proposed. The features include a middle point, length and angle of the line segment. Based on these features, appropriate standard patterns are selected. The feature distribution of the standard patterns are mapped onto that of the handwritten pattern. The feature mapping with structural constraints is proposed, which can provide flexible mapping and very fast convergence. The feature mapping results are estimated based on similarity between the distorted pattern and the mapped standard ones, convergence rate and deviation from the standard patterns. Computer simulation demonstrates distortion free feature extraction and flexible feature mapping.

I INTRODUCTION

Japanese Kanji characters have their structural meaning. They are composed of certain number of writing strokes. About 3000 characters are included in the first group of Japanese Industrial Standard (JIS), which are daily used. Totally, about 6000 characters are recommended by JIS to be used in real world. They have many similar structures. Structures themselves are also very complicated. For this reason, handwritten Japanese Kanji character recognition is inherently difficult subject.

Neural network approaches to pattern recognition are classified into the following categories. First, the distorted patterns are directly applied to the neural network. Topological features are extracted through the network. The pattern is recognized by matching it with standard patterns [1]. In the second method, some distortion invariant features are extracted by conventional methods, and these features are applied to multilayer neural networks, trained by supervised learning algorithms [2],[3]. Third method is a combination model of competitive learning and back-propagation, which is suited to large scale character recognition [4]. In any approaches, complicated networks and a long computing time are required.

In this paper, a new approach to handwritten Japanese Kanji character recognition is proposed. It consists of the cellular neural networks, for extracting features of line segments, and structure invariant feature mapping.

II PATTERN RECOGNITION SYSTEM

Figure 1 shows a block diagram for the proposed Japanese Kanji character recognition system. A distorted pattern is applied to the I-layer. It is skeletonized in the S-layer. In the L-layer, line segments are extracted. Vertical lines, horizontal lines and inclined lines with ± 45 degrees are extracted in the L_1, L_2, L_3 and L_4 networks, respectively. In the A-layer, the angle deviation is detected. In the TR-layer, the line segments are traced starting from the end points. The middle points and the lengths of the line segments are extracted. Each line segment is characterized by the above three kinds of features. The extracted line features are gathered on the F-layer. The networks used above are developed using cellular neural networks [5],[6].

Comparing the number of the line segments, which are specified with three kinds of features, appropriate candidate of the standard patterns are selected. The feature distribution of the standard pattern is mapped onto that of the distorted pattern, while maintaining topological structure [6],[7]. The mapping result is estimated based on three kinds of measures, similarity, convergence rate and deviation from the standard patterns.

Ⅲ LINE SEGMENT EXTRACTION

3.1 L_1 and L_2 Networks

Since the L_1 network can be replaced by the L_2 network, by exchanging row and column, the L_1 network is only described in this section.

Network Structure:

Since the network can be regarded as a matrix, a unit, which locates on the ith row and the jth column, is denoted by u(i,j). Furthermore, the input and output of u(i,j) are expressed by x(i,j) and y(i,j), respectively. Connection weights, used in the L_1 network, are defined as follows:

s: Self-loop of u(i,j).

a_{vk}: Bidirectional connection weights between u(i,j) and u(i,j-k). Two units, whose distance is k units, are connected by a_{vk}. It is independent from the coordinate (i,j), and is determined only by the distance k.

θ: Threshold level. u(i,j) is activated if its input is greater than or equal to θ.

The remaining connection weights are zero.

Network Dynamics:

The L_1 network state, which is a set of the unit outputs, is initially set to be the input pattern. The network changes its state as described in the following, resulting into the equilibrium state. In this sate, only vertical line segments with the specified minimum length can remain.

x(i,j) and y(i,j) are denoted by $x_{i,j}(n)$ and $y_{i,j}(n)$, respectively, in order to descries the state transition here.

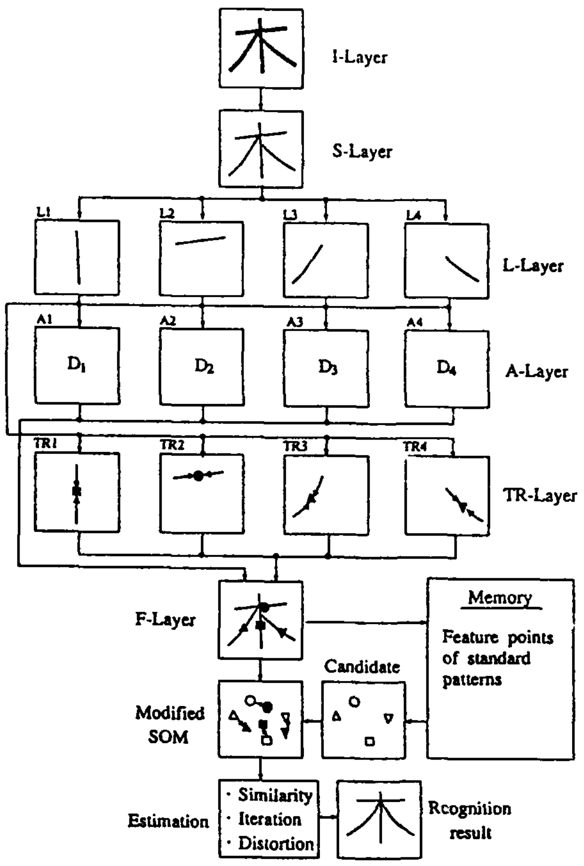


Fig.1 Block diagram of Japanese Kanji character recognition system.

$$y_{i,j}(0) = \begin{cases} 1, & \text{u(i,j) is included in the input pattern} \\ 0, & \text{otherwise} \end{cases} \qquad (1)$$

$$x_{i,j}(n) = s y_{i,j}(n-1) + \sum_{k} a_{vk}[y_{i,j-k}(n-1) + y_{i,j+k}(n-1)], \quad n \geq 1 \qquad (2)$$

$$\text{If } x_{i,j}(n) \geq \theta, \text{ then } y_{i,j}(n+1)=1 \qquad (3a)$$

$$\text{If } x_{i,j}(n) < \theta, \text{ then } y_{i,j}(n+1)=0 \qquad (3b)$$

Conditions for Extracting Line Segments:

Conditions for extracting lines segments, with the minimum length of m-units, are given in the following.

· Extract line segments, with the minimum length of m-units:

$$x_1(i,j) = s + \sum_{k=1}^{p} 2a_{vk} + \sum_{k=p+1}^{m-p-1} a_{vk} \geq \theta, \ p=0,1,2,\ldots,[(m-1)/2] \qquad (4)$$

· Reject short line segments:

$$x_2(i,j) = s + \sum_{k=1}^{p} 2a_{vk} + \sum_{k=p+1}^{m-p-2} a_{vk} < \theta, \ p=0,1,2,\ldots,[(m-2)/2] \qquad (5)$$

· Reject non-line segments:

$$x_3(i,j) = \sum_{k=1}^{p} 2a_{vk} + \sum_{k=p+1}^{m-p-1} a_{vk} < \theta, \quad p=0,1,2,\ldots,[(m-1)/2] \qquad (6)$$

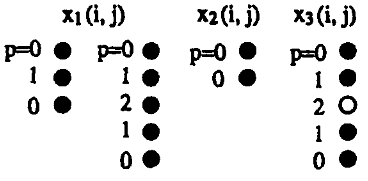


Fig.3 Relations between p and unit locations. m is chosen to be 3. ● active, ○ inactive.

In the above equations, [r] indicates the maximum integer not exceeding r. Figure 2 shows relations between p and the unit locations. The remaining connection weights are all zero.

3.2 L_3 and L_4 Networks

Connection weights are defined by,
L_3 Net b_{Rk}: Bidirectional connection weight between u(i,j) and u(i+k,j+k).
L_4 Net b_{Lk}: Bidirectional connection weight between u(i,j) and u(i-k,j+k).
The input of u(i,j) is expressed by

$$L_3 \text{ Net: } x(i,j) = sy(i,j) + \sum_k b_{Rk}[y(i-k,j-k) + y(i+k,j+k)] \tag{7}$$

$$L_4 \text{ Net: } x(i,j) = sy(i,j) + \sum_k b_{Lk}[y(i+k,j-k) + y(i-k,j+k)] \tag{8}$$

The same conditions can be derived for s, b_{Rk} and b_{Lk} as in the L_1 network.

3.3 Interaction among L_1, L_2, L_3 and L_4 Networks

In L_1 through L_4 networks, if the minimum length is chosen to be relatively long, curved lines and another inclined lines cannot be extracted. On the contrary, if the minimum length is chosen to be short, many non-line segments are extracted. In order to avoid such undesirable extractions, multistage extraction and interaction among the L_1 through L_4 networks are employed.

Step1: The vertical and horizontal lines with minimum length of m-units are extracted. m is chosen to be relatively large. The extracted line segments are removed from the original pattern. The remaining pattern is set on the L_1 and L_2 networks. The line segments with (m-1)-unit lengths are extracted, using the connection weights, which satisfy Eqs.(4)-(6). This step is repeated by decreasing the lengths. The extracted line segments are combined resulting the final vertical and horizontal line segments.

Step2: The inclined line segments (±45 degrees) with the minimum length of m-units are extracted. The extracted line segments are removed from the original pattern. By setting the remaining pattern on the L_3 and L_4 networks, the same line extraction with (m-1)-unit lengths is repeated. This step is repeated by decreasing the lengths. The extracted line segments are combined resulting the final inclined line segments.

Step3: If the extracted inclined line segments are completely included in the previous vertical or horizontal line segments, then they are removed.

IV ANGLE, MIDDLE POINT AND LENGTH EXTRACTION

4.1 Angle Extraction in A-Layer

Deviation from the standard angles, that is vertical, horizontal and slopes with ±45 degrees, is detected by using the following cellular neural network. Connection weights, defined in 3.1 and 3.2, are determined as follows:

A_1 Net (vertical):	$b_{R1} = -1$,	$b_{L1} = 1$	(9a)
A_2 Net (horizontal):	$b_{R1} = 1$,	$b_{L1} = -1$	(9b)
A_3 Net (+45 degrees):	$a_{V1} = 1$,	$a_{H1} = -1$	(9c)
A_4 Net (-45 degrees):	$a_{V1} = -1$	$a_{H1} = 1$	(9d)

The unit input is expressed as follows:

$$x(i,j) = a_{V1}[y(i,j+1) + y(i,j-1)] + a_{H1}[y(i+1,j) + y(i-1,j)] \\ + b_{R1}[y(i+1,j+1) + y(i-1,j-1)] + b_{L1}[y(i+1,j-1) + y(i-1,j+1)] \tag{10}$$

The angle deviation is given by

$$D = \frac{1}{2(N_U-1)} \sum x(i,j), \quad i,j \in \Omega_L \tag{11}$$

N_U is the number of units included in the line segments. Ω_L means a set of coordinates of the units. D represents the deviation from the standard angles. The standard angles are also normalized as follows:

Slope (-45 degrees):	-1
Horizontal line:	0
Slope (+45 degrees):	1
Vertical line:	2

Letting the normalized angle be A_0, the whole angle A is calculated as follows:

$$A = ((\, A_0 + D \,))_4, \quad ((\, n \,))_N \text{ means n modulo N.} \tag{12}$$

4.2 Middle Point and Length Extraction in TR-Layer

The middle point and the length of the line segment are obtained by tracing the line segments, starting from the end points. The connection weights are determined as follows:

$$w(i,j) = \begin{cases} W_{LT}, & i=j \\ 0, & i \neq j \end{cases} \tag{13}$$

$$a_{Vi} = a_{Hi} = b_{Ri} = b_{Li} = 1 \tag{14}$$

w(i,j) is connection weight from the ith unit in L-layer to the jth unit in TR-layer. The input of u(i,j) is expressed by

$$x(i,j) = W_{LT}y_F(i,j) + a_{Vi}[y(i,j+1)+y(i,j-1)] + a_{Hi}[y(i+1,j)+y(i-1,j)] + b_{Ri}[y(i+1,j+1)+y(i-1,j-1)] + b_{Li}[y(i+1,j-1)+y(i-1,j+1)] \tag{15}$$

$y_F(i,j)$ and y(i,j) are the outputs of u(i,j) in the L-layer and the TR-layer, respectively. At the first network transition, the state of u(i,j) in the TR-layer is determined by

$$y(i,j) = \begin{cases} 1, & x(i,j) \geqq \theta \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

At the following steps, the output is determined by

$$\text{If } x(i,j) \geqq \theta, \text{ then } y(i,j) = x(i,j) - \theta + 2 \tag{17}$$

By setting $\theta = W_{LT}+1$, the end points of the line segment are activated at the first step. In the following steps, the neighborhood of the activated units can be successively activated.

The middle point can be detected as the colliding point of two traces. Furthermore, the length of the line segment can be obtained as a sum of the outputs of the colliding units. The length is normalized by the total number of units included in the original pattern.

V PATTERN RECOGNITION BY FEATURE MAPPING

5.1 Mental Distortion

In the proposed method, standard pattern is mapped onto the distorted version, while maintaining topological structure. Because, the former is familiar to the human brain. Therefore, this mapping process can be regarded as mental distortion in human brain.

5.2 Candidate of Standard Patterns

Line segment features of standard patterns are extracted, and are stored in the memory. Candidate of the standard patterns are selected by comparing the number of the line segments characterized with three kinds of features.

5.3 Structure Invariant Feature Mapping

Kohonen's self-organizing map [8] is improved as follows:

(1)Feature points of the standard pattern are mapped onto those of the distorted pattern.
(2)Feature points are mapped onto the corresponding feature points.
(3)Feature points are selected in the variable ring shape region, in order to make it easy to find the corresponding feature.
(4)Feature points are selected from both patterns, in order to avoid double mapping and oscillation in the mapping process.

The proposed feature mapping process is described in the following.

(Step1) Selecting Feature Points:

The mapping is carried out on NxN grids. Feature points in both patterns are selected in the 1st outside region, whose coordinates are given by (1,j), (N,j), (i,1), (i,N), i,j=1~N.

(Step2) Selecting Corresponding Feature Point:

When p_K is selected first, q_m, which satisfies

$$\alpha \mid L_K - L_m \mid + \beta \mid A_K - A_m \mid < \varepsilon, \quad \alpha \text{ and } \beta \text{ are weighting factors} \tag{18}$$

is selected as the corresponding feature point. L_K, A_K and L_m, A_m are the lengths and angles of p_K and q_m, respectively. If several q_m satisfy this condition, one of them, locates closest to p_K, is finally selected.

p_K is shifted toward the selected q_m. On the other hand, when q_m is selected first, p_K, which satisfies the above conditions, is selected as the corresponding feature point. In this case, p_K is also shifted toward q_m.

(Step3) Neighborhood Constraints:

When p_K is shifted toward q_m, it's neighborhood are also shifted toward the same direction with shorter distance than that of p_K.

(Step4) Narrowing Ring Shape Region:

Feature points are selected in the 2nd outside region, whose coordinates are given by (2,j), (N-1,j), (i,2) and (i,N-1), i,j=2~N-1. Steps2 and 3 are repeated for all feature points in this region. After the region reaches the central point, the mapping process returns to Step1. The above processes are further repeated until the mapping converges.

5.4 Estimation of Mapping Results

Similarity:

The feature distribution P is assumed to be changed to P', after the mapping. In order to estimate similarity between P' and Q, the following error function is employed.

$$S = \frac{\Sigma\,[p'(n),q(m)]_1}{\Sigma\,[p'(n),q(m)]_1 + \Sigma\,[p'(n)]_2 + \Sigma\,[q(m)]_3} \qquad (19)$$

$[\,]_1$: The number of pairs of p'_K and q_m, which are mapped onto.
$[\,]_2$: The number of p'_K, which are not mapped onto the corresponding feature point.
$[\,]_3$: The numbers of q_m, onto which any p'_K are not mapped.
Therefore, perfect mapping yields S=1, otherwise S< 1.

Convergence Rate:

Feature point mapping from the 1st region to the central region is regarded as one iteration. A convergence rate is measured by the number of this iteration.

Variance from Standard Pattern:

Relative deviation between P' and P is estimated. Let (i,j) and (i',j') be the coordinates of elements in P and P', respectively. Average of translation (i_m, j_m) is given by

$$i_m = \frac{1}{N_L}\Sigma\,(i'-i), \qquad j_m = \frac{1}{N_L}\Sigma\,(j'-j) \qquad (20)$$

where N_L is the number of the line segments. The relative distortion is estimated by

$$V_i = \frac{1}{N_L}\Sigma\,(i'-i-i_m)^2, \qquad V_j = \frac{1}{N_L}\Sigma\,(j'-j-j_m)^2 \qquad (21)$$

VI SIMULATION

6.1 Japanese Kanji Characters

Handwritten Japanese Kanji characters have been dealt with. They are expressed using two-level values and 24× 24 dots. The first group of JIS Kanji characters (2965) are used for standard patterns.

6.2 Handwritten Kanji Character Recognition

Figure 3 shows an example of a handwritten distorted pattern, and it's skeletonized pattern. Figure 4 shows the extracted line segments in the L-layer. The minimum line length in the first step is chosen to be m=3, and the line extraction is repeated using m-1=2. Since some margin is employed for selecting the standard patterns, and topological structure is not taken into account, 16 different Kanji characters are selected. Among them, 「右」, 「石」, 「在」, 「左」, 「庄」 and 「戸」 have similar structure to that of the distorted pattern.

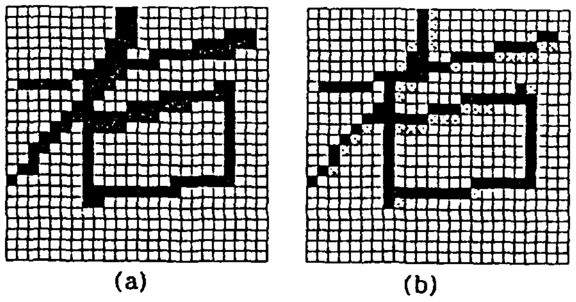


Fig.3 Example of distorted pattern 「右」.
(a) Input pattern
(b) Skeletonized pattern.

Figure 5 shows feature point distributions of the standard pattern 「右」 ◉, and the distorted version ■, and shifting directions.

The three measures, obtained by the feature mapping, are listed in Table 1. Since 「右」, 「石」, 「在」 have almost same topological structure, the similarities defined by Eq.(19) become S=1. Therefore, they cannot be distinguished based only on the similarity.

However, their differences are apparent based on the convergence rates and the variances. As a result, 「右」 can be recognized. Since the number of iteration is less than 20, the mapping process is very fast.

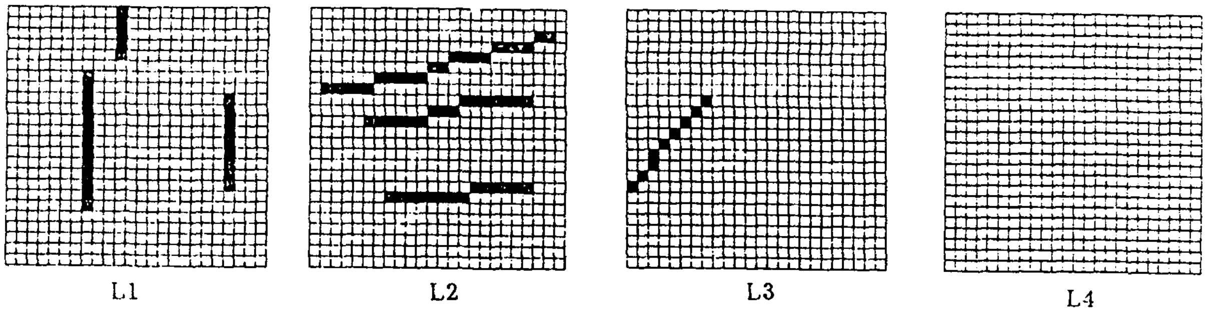


Fig.4 Extracted line segments of distorted 「右」 pattern.

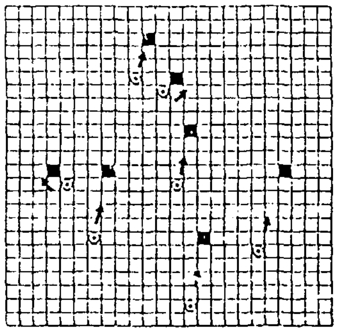


Fig 5 Feature point distributions, and shifting directions.

Table 1 Three measures and order of standard patterns, estimated by feature mapping.

Standard Patterns	Similarity S	Iteration	Variance Vi Vj	Order of Estimation
右	1.00	12	0.69, 3.39	1
石	1.00	15	0.82, 6.78	2
在	1.00	16	6.82, 6.24	3
庄	0.92	13	10.2, 4.14	4
左	0.92	13	8.22, 16.0	5
戸	0.83	11	0.56, 2.64	6

VI CONCLUSIONS

A new approach to handwritten Japanese Kanji character recognition has been proposed. The ideas behind the proposed are based on line feature extraction by the cellular neural networks and mental distortion by the structure invariant feature mapping. Computer simulation using about 3000 Kanji characters has demonstrated. Distorted patterns with scaling, translation, rotation and general distortion, can be recognized.

REFERENCES

[1]K.Fukushima and N.Wake, "Handwritten alphanumeric character recognition by the Neocognitron", IEEE Trans. on Neural Networks, vol.2, No.3, pp.355-365, May 1991.

[2]Y.Kimura, "Distorted handwritten Kanji character pattern recognition by a learning algorithm minimizing output variation", Proc. IJCNN'91,pp.I-103-I-106, Seattle, 1991.

[3]S.D.Hyman et al, "Classification of Japanese Kanji using principal neural network", Proc. IJCNN'91, pp.I-233-I-238, Seattle 1991.

[4]A.Iwata et al, "A large scale neural network "CombNET" and its application to Chinese character recognition", Proceedings of IJCNN'90, pp.83-86, Paris 1990.

[5]Y.Chigawa and K.Nakayama, "A pattern recognition model with pattern feature extraction and additional learning (in Japanese)", Proc. Technical Meeting of IEICE of Japan, vol NC91-12, pp.87-94, May 1991.

[6]Y.Chigawa, Y.Mori and K.Nakayama, "Handwritten Kanji character recognition by using segment feature extraction and modified SOM (in Japanese)", Proc. Technical Meeting of IEICE of Japan, vol.NC92-13, pp.17-24, May 1992.

[7]K.Nakayama, Y.Chigawa and O.Hasegawa, "Handwritten alphabet and digit character recognition using feature extracting neural network and modified self-organizing map", Proc. IJCNN'92, vol. IV, pp.235-2240, Baltimore 1992.

[8]T.Kohonen, "Self-Organization and Associative Memory", 3rd ed., Springer-Verlag, 1989.

Nonlinear CNN Clonning Templates for Image Thicking

Second International Workshop on Cellular Neural Networks and Applications CNNA'92

Stanisław Jankowski and Ryszard Wańczuk
Institute of Electronics Fundamentals
Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
Tel. (+48 22) 25 75 44; Fax: (+48 22) 25 23 00;

Abstract

The idea of nonlinear and delay-type clonning templates which enable to increase essentially the efficiency of the CNN was described in [1]. This paper is devoted to the image thicking templates which enable to improve the quality of distored images. Especially it was found that these type of the CNN can be successfully used in preprocessing of handwritten characters in order to recognize them by the four CCD operators method [4]. Three various thicking templates are considered (demonstrated) : two examples of nonlinear clonning templates and one delay-type template.

1. Introduction

The cellular neural networks (CNN) are effective tools for image processing due to their simple operation mode and regular configuration [2,3]. The number of the CNN applications has been rapidly growing. It was shown that these networks can be efficiently used for the handwritten character recognition by the connected component detectors (CCD) method [4]. It is observed that primary images of handwritten characters obtained from the scanner have irregular noisy contours. This fact may decrease the efficiency of the recognition by the CCD method.

The contours of figures can be improved (i.e. smoothed, regularized) by thicking. Three various CNN algorithms of thicking are presented in this paper:

- nonlinear clonning template I - simple thicking;
- nonlinear clonning template II - which enables to obtain the figure contour and additionally thicking by one pixel after logical "OR" operation with the input image;
- delay-type template which gives efficient smoothing and thicking by local averaging.

2. Thicking clonning templates

2.1. The nonlinear template I for contour thicking may be expressed as follows using the usual CNN notation:

$$A = \begin{bmatrix} 1 & d & 1 \\ d & 5 & d \\ 1 & d & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad I = 9 \tag{1}$$

where $d = 0.5 \cdot (Y[k,l])^2 - (Y[k,l] \cdot Y[i,j])$.

Due to this rule the contour of an object can be increased by one cell in all directions, as shown in Fig. 1. It does not require any special initial and boundary conditions - it is sufficient to impose -1 at the boundary and to take the image itself as an initial condition. The operation mode takes only several iterations while simulating on a computer.

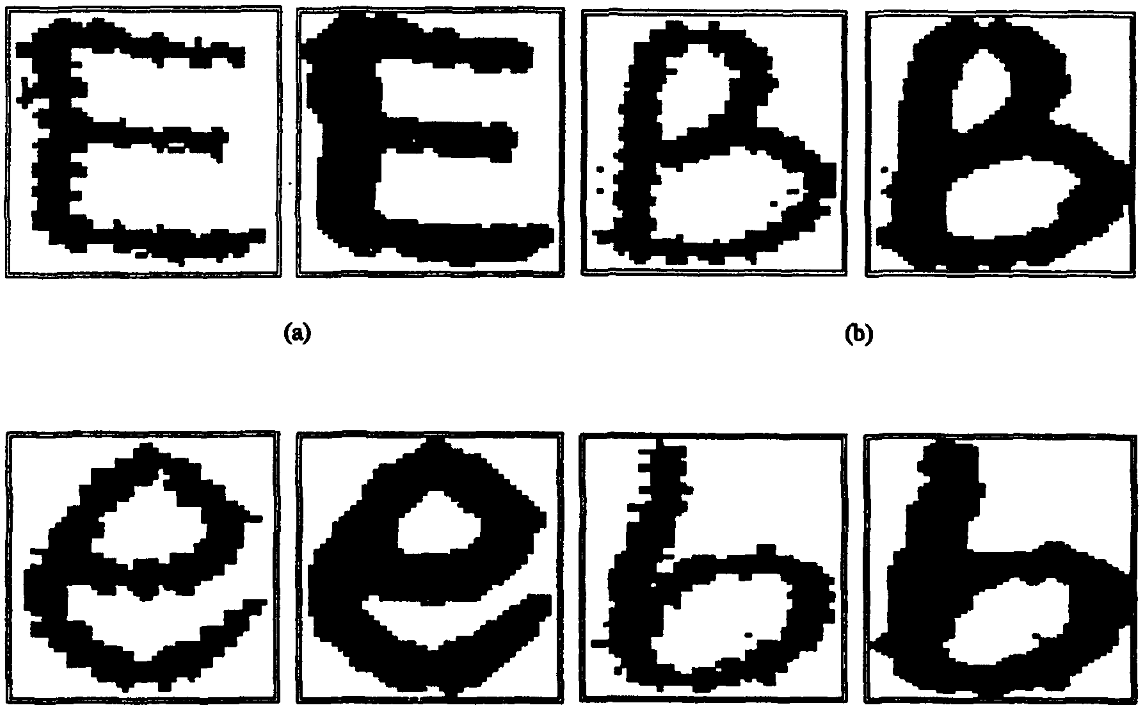


Fig. 1: Demonstration of the thicking template. (a)(b)(c)(d) Input and output images.

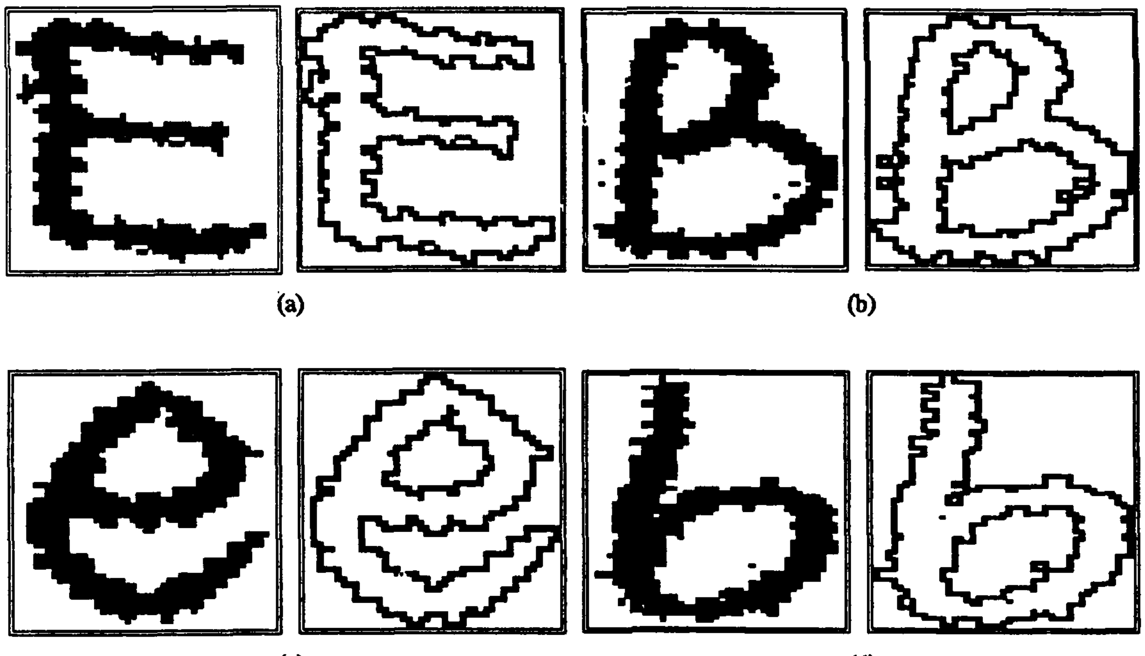
As can be seen from Fig.1. the clonning template (1) improves the contours of input images by smoothing. This rule is characterized by good stability properties.

2.2 The nonlinear clonning template II for drawing contours is described below:

$$A = \begin{bmatrix} 0 & d_1 & 0 \\ d_2 & 0 & d_2 \\ 0 & d_1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 4 & 4 & 4 \\ 4 & -30 & 4 \\ 4 & 4 & 4 \end{bmatrix} \qquad I = -5 \tag{2}$$

where $d_1 = (Y[k,l])^2 - 0.5 \cdot (Y[k,l] \cdot Y[i,j])$,
$d_2 = -0.5 \cdot (Y[i,j] \cdot Y[k,l])$.

This rule enables to draw the contour of a given object in all directions and the contour is shifted by one, as shown in Fig. 2. The operation mode takes few iterations (or time costants in analog circuit).It is possible to use this rule many times to obtain multiple contours of figures.


(a) (b)

(c) (d)

Fig. 2: Demonstration of the drawing contour template. (a)(b)(c)(d) Input and output images

In order to obtain the image thicker by 1 pixel it is necessary to perform the logical "OR" of the computed contour and the input image. This operation can expressed by:

$$A = 1 \qquad B = 1 \qquad I = 1 \tag{3}$$

The examples of results are illustrated in Fig. 3.

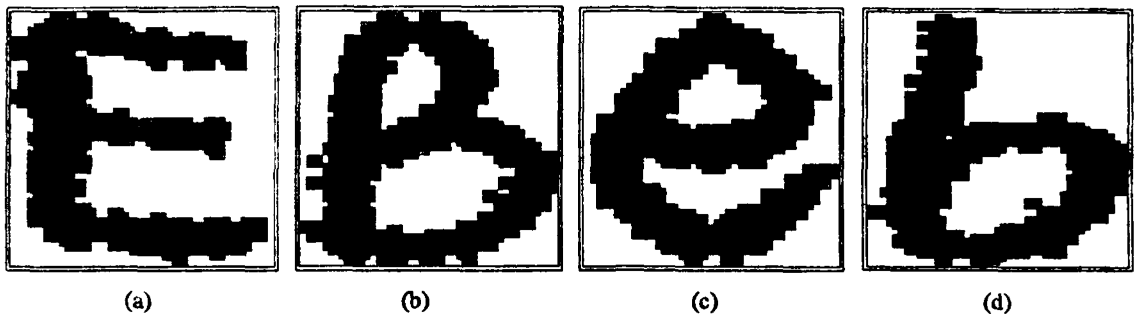


Fig. 3: Demonstration of the thicking through drawing contour and logical "OR" with input image. (a)(b)(c)(d) Output images.

2.3 Delay-type clonning template for improvement of handwritten character image at the output of a scanner has the following form:

$$A = \begin{bmatrix} 0 & 0.25 & 0 \\ 0.25 & 0 & 0.25 \\ 0 & 0.25 & 0 \end{bmatrix} \quad B = 0 \quad \begin{matrix} I = 0.0 \\ t1 = 6 \end{matrix} \tag{4}$$

$$A^{t1} = 1 \quad B^{t1} = 0 \quad I^{t1} = 0.5 \quad t2 = 9$$

$$A^{t2} = 3 \quad B^{t2} = 0 \quad I^{t2} = 1.5$$

The template improves the quality of the character by noise removing and by smoothing the irregularities at the border, as presented in Fig. 4. The image itself is used as the initial condition and all boundary cell states are equal to -1.

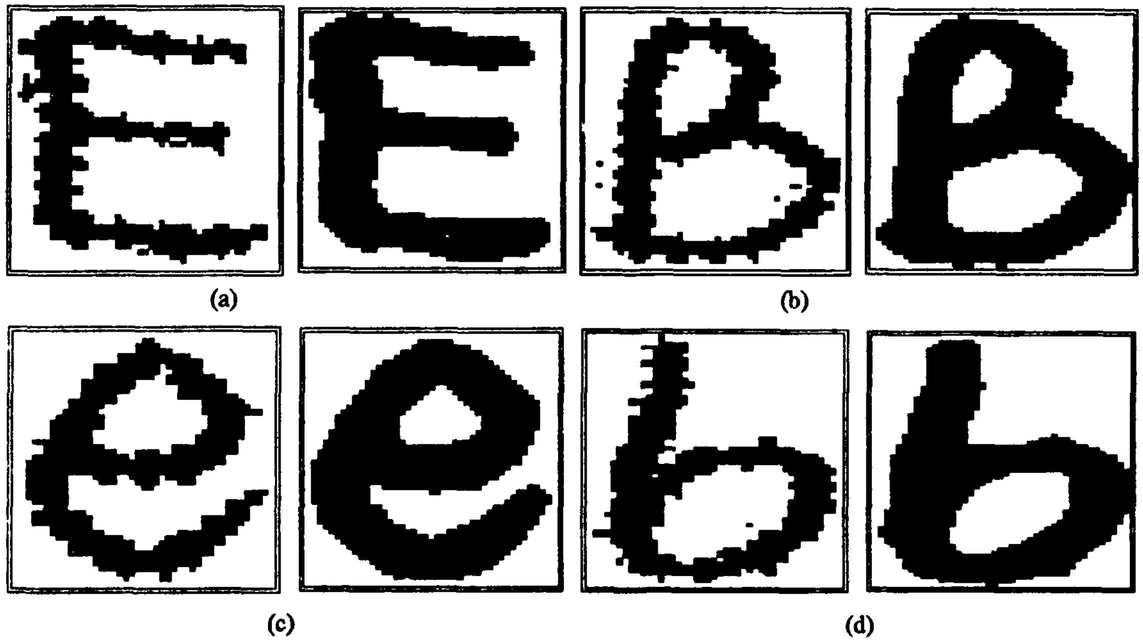


Fig. 4: Demonstration improving quality of handwritten characters. (a)(b)(c)(d) Input and output images after once use of template (3).

This template operates in three steps.

- STEP I : six iterations of the rule determined by A, B and I;
- STEP II : three iterations defined by the operators

$$A' = A + A^{t1} \quad B' = 0 \quad I' = I + I^{t1}$$

- STEP III: a sequence of iterations obtained in the following way:

$$A'' = A' + A^{t2} \quad B'' = 0 \quad I'' = I' + I^{t2}$$

This clonning template may be modified by changing the template parameters: the delays t1 and t2 and current values. The STEP I of the algorithm causes the averaging of the input image - the contour becomes fuzzy and the networks tends to the grey-scale state. This effect increases in function of time. The averaging stops by using the STEP II - the network tends to the stable state -1 (white image) in the whole area of the considered image. Then the use of the STEP III makes the stable output, thicker than the input and with the smooth contour. Increasing of the delays t1 and t2 amplifies the thicking effect (up to hole-filling), as can be stated from Fig. 5.

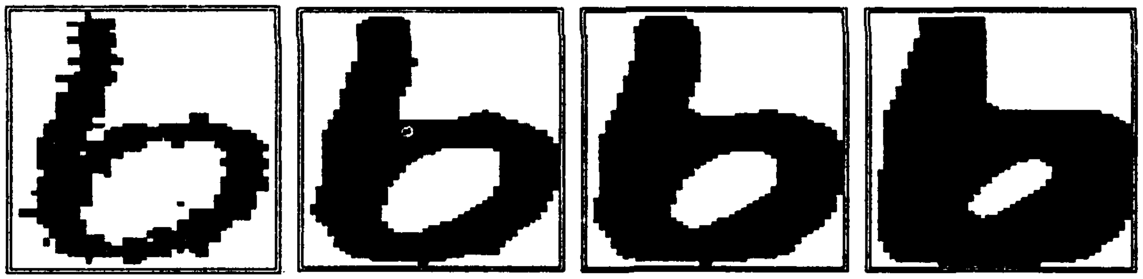


Fig. 5: Improving quality of handwritten characters. (a) Input image. (b) Output image : t1=6,t2=9. (c) Output image : t1=8,t2=13. (d) Output image : t1=11,t2=32.

3. Conclusions

Three new clonning templates for the CNN are described in this paper. Their aim is to perform the improvement of noisy image with highly irregular contour by the image thicking and contour smoothing. These operations cause the regularization of hand-written character images after scanning and make character recognition more efficient by using, e.g. the connected component detector method. All presented templates: nonlinear I and II as well as the double-delay-type template are stable and useful. The simplest is the nonlinear I template. The nonlinear II template consists in obtaining the outer contour of the figure and thicking by the "OR" operation with the input image. The delay-type operation can be modified by changing its delay parameters. The results of operation of presented templates are confronted in Fig. 6.

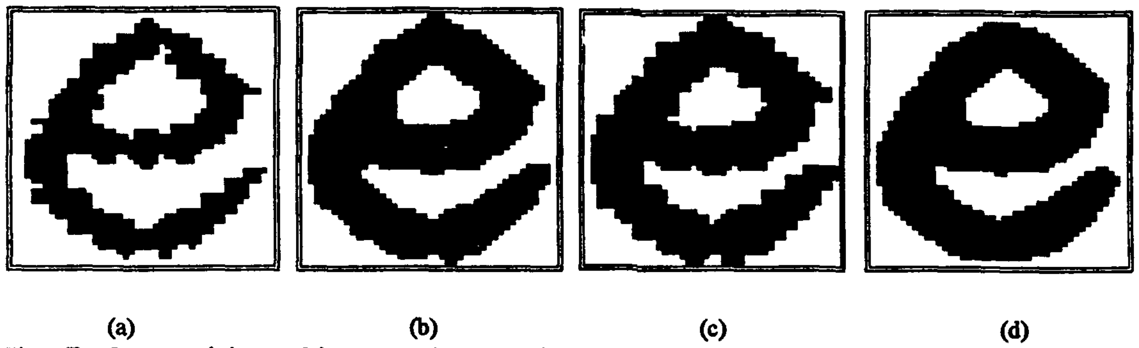


Fig. 7: Comparision of input and output images after thicking by following templates. (a) Input hand-written character. (b) Thicking by nonlinear template I. (c) Thicking by nonlinear template II. (d) Thicking by delay-type template.

REFERENCES

[1] T.Roska and L.O.Chua: 'Cellular neural networks with nonlinear and delay-type template elements',*Proc.IEEE CNNA-90*,12-25,1990.

[2] L.O.Chua and L.Yang: 'Cellular neural networks: Theory',*IEEE Transactions on Circuits and Systems* 35,1257-1272,1988.

[3] L.O.Chua and L.Yang: 'Cellular neural networks: Applications',*IEEE Transactions on Circuits and Systems* 35,1272-1290,1988.

[4] T.Yokohama, H.Suzuki, Y.Matsushita, T.Matsumoto and L.O.Chua: 'Non-Symetric CNN Templates for Image Processing',*Proc.ECCTD-91*,10-19,1991.

An experimental system for optical detection of layout errors of printed circuit boards using learned CNN templates

P.Szolgay ,I.Kispál$^+$ *and T.Kozek*

Dual and Neural Computing Systems Laboratory

Computer and Automation Institute Hungarian Academy of Sciences, P.O.B.63, H-1502 Budapest, Hungary

+ Department of Information Technology, Faculty of Engineering, University of Veszprém, Egyetem u. 10, H-8201 Veszprém, Hungary

Abstract

Cellular neural networks (CNNs) are considered here as cellular analog programmable multidimensional processing arrays with distributed logic and memory. The interconnecting weights between the neighbouring processing elements are defined by the template values.In the paper a new systematic way is presented to find robust templates. Using the new learning algorithm some templates were found for a CNN based layout design rule checking algorithm. The algorithm has been tested in our experimental system with real life examples. A typical design rule checking of a 432x164 pixels area takes 8s computation time on our CNN hardware accelerator board.

1 Introduction

Cellular neural networks [1] (CNNs) are considered here as cellular analog programmable multidimensional processing arrays with distributed logic and memory [11].

The interconnecting weights between the neighbouring processing elements, cells, are defined by cloning template values. Besides the linear cloning templates, the nonlinear and delay type templates proved to be substantially widening the applications [3]. A cloning template defines the transformation of an input- and initial state signal-array (e.g. image) to an output signal-array. Some templates have been found by a "cut-and-try" method, but recently, systematic template learning algorithms were reported [5],[7],[10],[13]. The learning process presented here is based on simplex optimization method. The learned templates are robust, i.e. they are less sensitive to the changing of the values of their elements.

The layout rule checking of a printed circuit board documentation or a manufactured board is time consuming. Detection of layout errors requires mostly local geometrical information, hence the task well suits the CNN paradigm. The typical layout errors on a PCB artwork film or on a manufactured PCB are as follows: (i) wire width is smaller than a given value, even a wire may be broken, (ii) the isolation on the layout is smaller than a given value, even a short circuit may be produced, (iii) the brake of a pad, (iv) fleck or pinhole on a wire, and (v) the misalignment of the pads to the holes. A key problem is the detection of the minimal line width violations. It can be shown that most of the layout error detections can be transformed to this problem.

Considering the different CNN hardware implementations [2],[6] it can be seen that for large images (over 100 thousand pixels) the digital multiprocessor add-on-board is the only solution. Our new CNN-HACM [4] board provides a 1 million cell space at 1.5 μs/cell/iteration speed. Here, a low cost experimental solution will be shown based on our CNN-HACM hardware accelerator board to solve the layout design rule checking problem. The most attractive solution is, naturally, the analog VLSI solution. An IBM add-on-board with a 0.4mm minimal wire width (with 200 dot/inch resolution) can be processed in 2 parts using our new CNN-HACM accelerator board.

Template learning methods are introduced with novel properties in Section 2. New templates and subroutines [13] are developed in Section 3 to solve the above basic layout detection problem. Real-life examples are summarized in Section 4. Finally, in Section 5 the theoretical and the practical limitations are discussed.

2 Learning algorithm for CNN

Template design or learning algorithm for cellular neural networks means a procedure which is able to find the template for a given operation. The operation is specified by the two input signal arrays $S_1(ij)=v_{uij}$ and $S_2(ij)=v_{xij}(0)$ and the desired output $S_0(ij)=v_{yij}(y)$.

Learning algorithms published so far [5],[7],[10],[13] are based on a system of inequalities and by solving them, linear templates are obtained. Consider the state equation in [1.a]. Assuming that the equilibrium points of the network are stable, i.e. for $t\to\infty$ $dv_{xij}/dt=0$, it follows that

$$v_{xij}(\infty) = \sum_{kl \in N_r(ij)} A_{ij;kl} \cdot v_{ykl} + \sum_{kl \in N_r(ij)} B_{ij;kl} \cdot v_{ukl} + I_{ij}$$

where $v_{xij}(\infty)$ is the settled state. Assuming further that the desired output is binary, i.e. $v_{yij}(t)=\pm 1$ for $t\to\infty$ and setting $v_{yij}(\infty)$ to the desired output $S_0(ij)$, because of [1.a], we get

$$\sum_{kl \in N_r(ij)} A_{ij;kl} \cdot v_{ykl} + \sum_{kl \in N_r(ij)} B_{ij;kl} \cdot v_{ukl} + I_{ij} \geq 1 \text{ if } v_{yij}(\infty)=1,$$

$$\sum_{kl \in N_r(ij)} A_{ij;kl} \cdot v_{ykl} + \sum_{kl \in N_r(ij)} B_{ij;kl} \cdot v_{ukl} + I_{ij} \leq -1 \text{ if } v_{yij}(\infty)=-1.$$

To assure stability the symmetry of the feedback template A is provided. To gain a more efficient template the following inequalities can be added to the above system providing a shorter and possibly monotone transient:

$$\sum_{kl \in N_r(ij)} A_{ij;kl} \cdot v_{ykl}(0) + \sum_{kl \in N_r(ij)} B_{ij;kl} \cdot v_{ukl} + I_{ij} - v_{xij}(0) \geq 0$$

$$\text{if } v_{yij}(\infty)=1 \text{ and } v_{xij}(0)<1,$$

$$\sum_{kl \in N_r(ij)} A_{ij;kl} \cdot v_{ykl}(0) + \sum_{kl \in N_r(ij)} B_{ij;kl} \cdot v_{ukl} + I_{ij} - v_{xij}(0) \leq 0$$

$$\text{if } v_{yij}(\infty)=-1 \text{ and } v_{xij}(0)>-1.$$

In [5] relaxation methods were used to find a solution of the system of inequalities. Some useful templates had been found, but in many cases changing the template elements by a few percent, the dynamic behaviour of the network changed as well and the template no longer performed the desired operation. To overcome this difficulty, a different algorithm was proposed [7] which is discussed in short here.

A learning algorithm for gaining robust templates

To obtain robust templates, we have to find not only an arbitrary solution of the inequality system, but to find an optimal one in the sense of being far inside the solution domain. The robustness of a template with respect to changing its elements depends on how far inside the parameter space it is from the boundaries defined by the inequalities. The greater the distance, the more robust the template, and the transient gets faster, as well.

To characterize a point in the parameter space a cost function is constructed whose global minimum is as far as possible from all of the boundaries of the solution domain. Locating the minimum of the cost function by some optimization method, a template with the desired properties can be gained.

To find this optimum the simplex optimization algorithm has been applied. If the domain of solutions in the $(2r+1)^2+1$ dimensional parameter space (the number of template elements with neighbourhood size r) is closed, then the "mass centre point" of the domain should give a robust template. In that case the cost function is defined as

$$\text{cost}(T) = \sum_i \delta(T, p_i)$$

where $\delta(T, p_i)$ denotes the signed distance between the point in the parameter space corresponding to the parameter vector (or template) T and the plain defined by the ith inequality. It is easy to see that this cost function is unimodal, that is, the optimization algorithm finds the global minimum without difficulty. There is no guarantee, however, that the domain of solutions is closed. In fact, in most cases the solution domain is open and the optimum lies in the infinity. To overcome this problem and to provide a controllable robustness for the templates, the following learning method has been used.

1. First, the simplex algorithm, which works with a set of parameter vectors, is initialized with templates having symmetric feedback A. The algorithm then quarantees that all parameter vectors remain in the subspace defined by the symmetry constraints.

2. In the first optimization step the simplex algorithm is run to bring all parameter vectors inside the, possibly open, solution domain. It stops when all the points corresponding to the parameter vectors are at least $\epsilon > 0$ far from the boundaries.

3. In the second run a different cost function is applied to bring the parameters into the feasibility range while the condition acquired in step 2 is maintained.

This method gives templates with a prescribed robustness with respect to changing their parameters. This means that the template elements can be slightly modified without changing the behaviour of the system. Alteration of the elements has no effect on the operation performed by the template at least until the absolute sum of the modifications remains under ϵ.

Limitations of the proposed learning algorithm

Learning algorithms based on the above system of inequalities have strict limitations. Only binary output templates with symmetric feedback can be generated. Besides, no information about the dynamics of the actual transient is considered. Consequently, templates with local dynamics can be learned only.

Recently a new learning algorithm [14] based on genetic search was reported. According to the experiences a broader class of template learning problem can be solved by using this algorithm.

3. The experimental system for PCB layout error detection

The main steps of PCB layout error detection are as follows,

(i) the optical input of the layout (in our experimental system we use a scanner input),

(ii) the transformation of the gray level image of layout to a black and white one,

(iii) perform the different layout error detection algorithms.

3.1 Setting the appropriate parameters of the optical scanner

Using an Epson GT4000 scanner [9], scanning was tested the typical classes of printed circuit boards and PCB artwork films. The point here is to have such a scanned gray scale area where the layout figures can easily be identified in the background isolation area. The resolution of the scanning should be at least 3-4 times finer than the smallest details of the scanned object.

3.2 The black and white image generation

The next step is to find a black and white picture from the gray scale one. Here the average, the thresholding, or some nonlinear filtering templates can be used.

3.3 The critical algorithms of the design rule checking of printed circuit board documentation by using cellular neural networks

A crucial problem of the layout checking means the detection of violations of the minimal line width and the minimal separation width. The second problem can be transformed to the first one. The pixels of lines or pads are considered black and the area between them white. If we were able to solve the problem of minimal line width violation detection on a positive picture then on a negative picture, the minimal separation violation could be detected by the same method.

The minimal line width violation detection of rectilinear layout

Here a sequence of image transformations will be shown to determine the places where the line width requirement were not met. The sequence of image transformations in CNN can be done by a sequence of templates.

The horizontal and the vertical line width checkings are performed separately and the faults are logically OR-ed. The minimal line width is first supposed to be 2 pixels, and later an algorithm is shown for N pixel minimal line width.

<u>The vertical wire case</u>

First, let us suppose that we have 2 pixels wide lines and these lines in some cases are thinner than 2 pixels. We have to find these places. Using our template learning program [7] with $\epsilon = 1.0$, the result is as follows.

$$A=\begin{bmatrix} -0.11 & 0.43 & -0.11 \\ 0.43 & 0.06 & 0.43 \\ -0.11 & 0.43 & -0.11 \end{bmatrix} \quad B=\begin{bmatrix} 0.19 & -0.04 & 0.19 \\ -2.92 & 2.40 & -2.92 \\ 0.19 & -0.04 & 0.19 \end{bmatrix} \quad I= -4.96$$

The template finds all pixels which are on a vertical line with one pixel width. Knowing that this template has been found with $\epsilon = 1.0$ value, the learned template could be rounded as follows:

$$A=\begin{bmatrix} -0.1 & 0.4 & -0.1 \\ 0.4 & 0.0 & 0.4 \\ -0.1 & 0.4 & -0.1 \end{bmatrix} \quad B=\begin{bmatrix} 0.2 & 0.0 & 0.2 \\ -3.0 & 2.5 & -3.0 \\ 0.2 & 0.0 & 0.2 \end{bmatrix} \quad I = -5.0$$

<u>The horizontal wire case</u>

To detect the errors in horizontal wires, the same template can be used as in the vertical case, however, the template B has to be rotated by 90 degrees.

<u>The general minimal line width violation detection</u>

The next problem is to find the places where a line is thinner than N pixels. To solve this problem, we have to find some algorithms containing several templates (CNN subroutines). The algorithm uses peeling templates which remove pixels from either sides of black objects. The following template turns white all black pixels having white pixels on their left side:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 3 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad I = -5$$

Templates for peeling from right side can be generated by rotating the template B above.

The CNN subroutine [13] for finding lines thinner than N pixels in vertical (horizontal) direction is as follows. Two images are used, namely a temporary "TEMP" and a result "RESULT".

```
- Load the input picture and save it in TEMP
- Find one pixel thin lines and save the result in RESULT
- Repeat N-2 times:
        - Load TEMP
        - Peel 1 pixel from left (up) or right (down) (best if
        alternating)
        - Save in TEMP
        - Find one pixel thin lines and OR the result in RESULT
```

Remarks:

(i) This algorithm has to be run twice to find both the horizontal and vertical errors.

(ii) It can be seen that the number of iterations depends on the minimal wire width to be detected.

(iii) The input picture and the initial state are supposed to be equal.

(iv) Here an additional picture has to be used to store the results (RESULT). It is crucial, in view of speed, how many picture changes are needed and how many additional pictures have to be stored. These are the parameters which determine the complexity of an analog or a dual algorithm.

After error detection phase the very short (<d) errors should be removed, supposing they are not layout errors but scanning errors. By the next template the isolated black points are removed.

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0.5 & 0 \\ 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & 0 \end{bmatrix} \qquad I = 0$$

4. Layout error detection experiences based on nontrivial examples

Experiment 1

A part of an artwork film of a multilayer PCB with 0.254mm minimal separation width was considered and the places have to be detected and recorded where the separation is smaller then this given value. The size of the picture was 362*648 pixels. In Figure 1a and Figure 1b the scanned binary input and the detection the errors of vertical separation can be seen. It took 8s processing time (and an additional 20s were necessary to load the input picture and the initial states) by using our CNN-HAC (v. 2.0) board [4].

Experiment 2

The gray level input of a part of a manufactured printed circuit board (603x309 pixels) is shown in Figure 2a. First, a black-and-white picture was generated (see Figure 2b) by using a nonlinear filtering and the average template. The minimal vertical line width violation (where the line width is smaller then 3 pixels) is detected. The result of the detection is shown in Figure 2c. The algorithm was used and the processing time with the CNN-HAC board [4] was 10s (an additional 50s were necessary to load the input and the initial states and to draw the results on the screen).

5. Theoretical and practical limitations

An important question is: in how many parts a real problem can be processed. Its running time consequence was shown previously. An IBM PC add-on-board with a 0.4mm minimal wire width (in 200 dot/inch resolution) can be processed in 2 parts by using our new CNN-HACM accelerator board.

With the proposed method the short circuits cannot be detected owing to the fact that it is not a local error. Even if we could give a node information to a pixel, it needs an additional storage requirement which can be 8 bits/pixel or more. Only the thin short circuits are detected by the minimal line width violation algorithm.

Acknowledgements

This work has been supported by the National Committee for Technological Development grant No 00052/910308. The authors would like to thank for the helpful discussions with Prof. Tamás Roska.

References

[1.a] L.O.Chua and L.Yang, "Cellular neural networks: Theory", IEEE Trans. on Circuits and Spystems, Vol.35, pp. 1257-1272, 1988.

[1.b] L.O.Chua and L.Yang, "Cellular neural networks: Applications", IEEE Trans. on Circuits and Systems, Vol.35, pp. 1273-1290, 1988.

[2] J.M.Cruz and L.O.Chua, "High-speed high-density CMOS CNNs", Memorandum No UCB/ERL M91/28, University of California Berkeley, 1991.

[3] T.Roska and L.O.Chua,"Cellular Neural Networks with Nonlinear and Delay-type Template Elements",Proc. of the IEEE CNNA-90, pp.12-25. 1990.

[4] T.Roska, G.Bártfai, P.Szolgay, T.Szirányi, A.Radványi, T.Kozek, Zs. Ugray and A Zarándy, "A Hardware Accelerator Board for Cellular Neural Networks: CNN-HAC",Proc. of the IEEE CNNA-90, pp.160-168. 1990.

[5] F.Zou,S.Schwarz and J.A.Nossek," Cellular neural network design using a learning algorithm, Proc. of the IEEE CNNA-90, pp.73-81. 1990.

[6] T.Roska and P.Szolgay, "A comparison of various cellular neural network (CNN) realizations - a review" Proc. of the 2nd Int. Conf. on Microelectronics for Neural Networks, pp. 423-431, 1991.

[7] P. Szolgay and T. Kozek, "Optical detection of layout errors of printed circuit boards using learned CNN templates", DNS-10-1991. MTA-SzTAKI Budapest, 1991.

[8] ED-Scan for EPSON-Colorscanner documentation, EPSON Inc.

[9] "Dual CNN software library" (ed. by T.Roska, A.Radványi, T.Kozek,T.Boros) DNS-7-1991. MTA-SzTAKI Budapest, 1991.

[10] S.Schwarz and W.Mathis, "A design algorithm for cellular neural networks", Proceedings of the 2nd Int. Conf. on Microelectronics for Neural Networks, pp. 53-59, 1991.

[11] T.Roska and L.O.Chua, " The dual CNN analog software - a programmable analog, nonlinear, dynamic, 3D computing array plus logic to form a multi-screen theater on silicon" DNS-2-1992. MTA-SzTAKI Budapest, 1992.

[12] T.Roska, A.Radványi and A Zarándy, " Dualcomp, Dual CNN compiler to CNN-HAC1 board", DNS-13-1992. MTA-SzTAKI Budapest, 1992.

[13] H.Magnussen and J.A.Nossek,"Towards a learning algorithm for discrete-time cellular neural networks", TUM-LNS-TR-92-5, Techn. Univ. Munich, 1992.

[14] T.Kozek,T.Roska and L.O.Chua,"Genetic algorithm for CNN template learning", UCB/ERL M92/82 University of California, Berkeley, 1992.

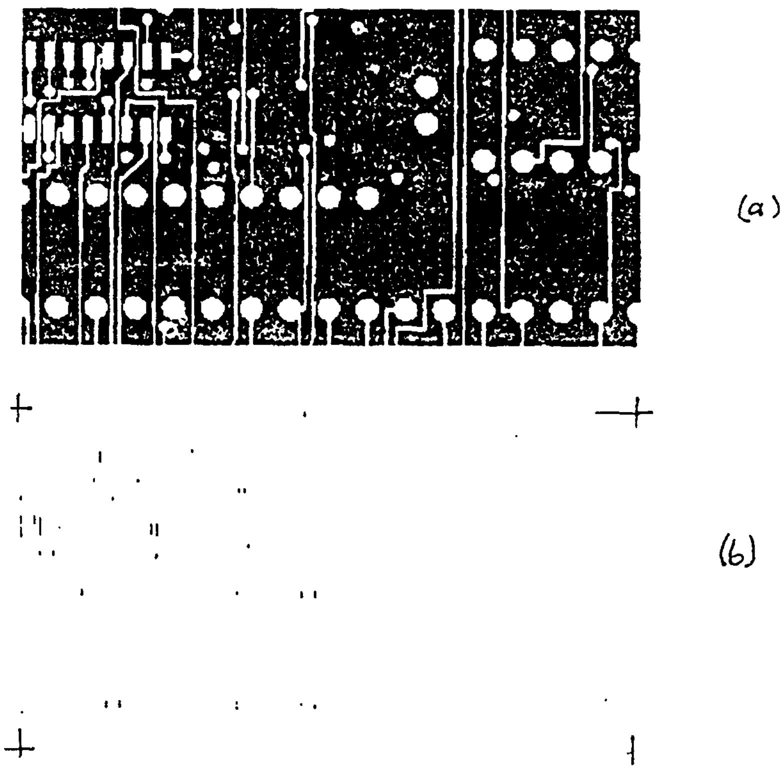


Figure 1 The input (a) and the result (b) of vertical minimal separation width violation detection on a PCB artwork film

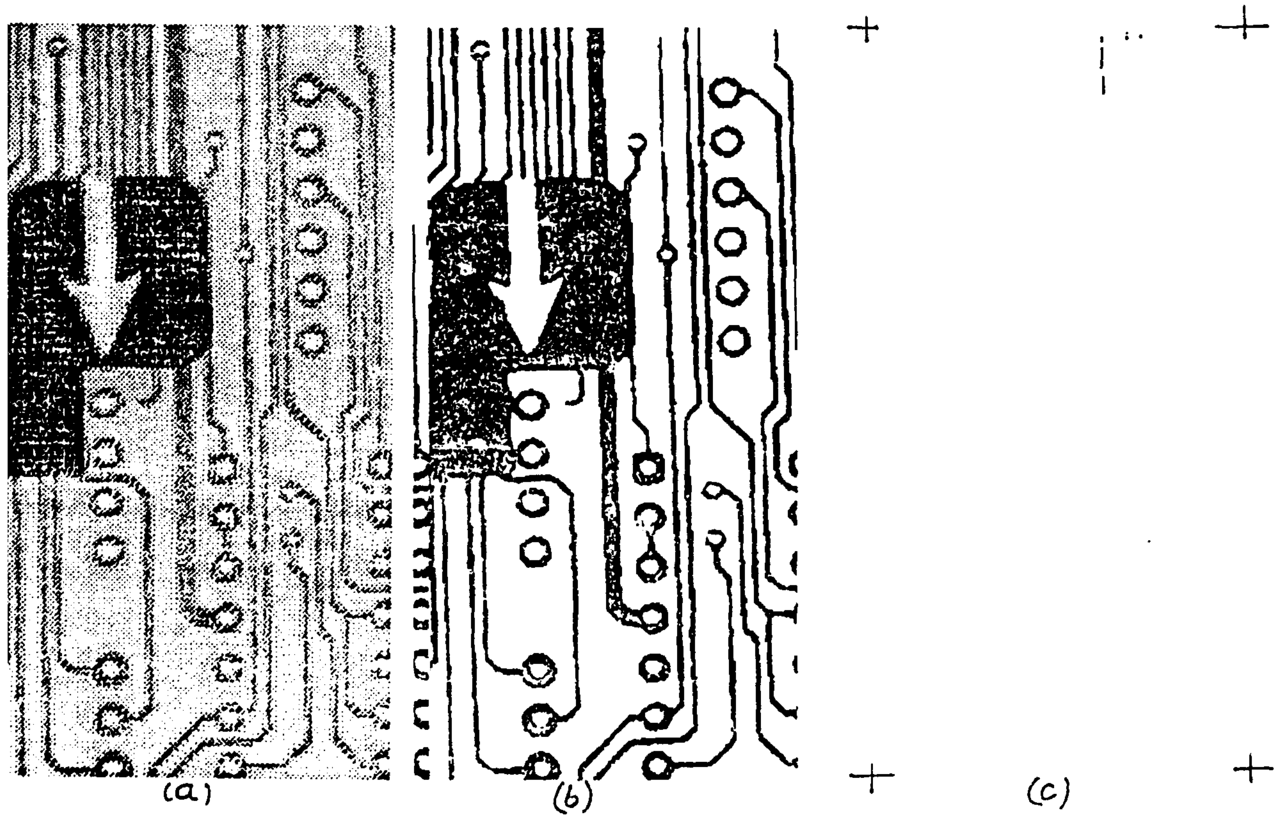


Figure 2 A part of a manufactured multilayer PCB was tested (a), the black and white picture of the oard (b) and the vertical line width violations on the board (c) are shown.

On the Application of Cellular Automata to Image Thinning with Cellular Neural Network

Dao-heng YU†, ††, Chun-ying HO†, Xiang YU†††, and Shinsaku MORI†

†Dept. of Electrical Engineering, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223 Japan
Tel. +81-45-563-1141 Ext. 3319 Fax. +81-45-563-2773

††Dept. of Radio Electronics, Peking University, China

†††Dept. of Automation, Beijing Sciences & Technology University, China.

Abstract

We are studying one class of new Cellular Automata (called B-rule in this paper). We have discovered rule B 132 can be exploited in the design of Cellular Neural Network (CNN) for binary image thinning. In this paper, we introduce B-rule 132 and the method of the binary image thinning with CNN.

Intruduction

Cellular Neural Network (CNN) [1-2] is a class of novel analog nonlinear circuit which allows real-time signal processing by using only local interconnections of neighboring cells. In this paper, we consider a class of new Cellular Automata (CA) which is called B-rule. We have discovered that the rule B_{132} can be exploited in the design of CNN for binary image thinning. In this context, the characteristics of one-dimensional (1-D) B-rule CA is introduced. Then we discuss the approach and results of employing these rules to binary image thinning by using CNN.

The relations between CA and CNN linear cloning templates have been addressed in [3]. In [4], the relations among CNN with CA and Systolic Array (SA) have been analyzed. If the operations of the system are purely logical and involve only a few bits, the classical CA of John Von Neuman and its recent variants are ideal tools. The thinning algorithms by using two-dimensional (2-D) CA have been discussed rigorously in [5] while in [6], the system proposed needs a complex eight-plane approach of CNN to accomplish. However, the authors in [6] have not mentioned the methodology to determine the values of the cloning templates. In the studies of a new class of 1-D CA, we have discovered that rule B_{132} (to be described later) are useful for the design of CNN for image processing. Henceforth, we shall refer 1-D CA simply as CA.

B-rule Cellular Automata

CA are sufficiently simple to allow detailed mathematical analysis, while on the other hand, complex enough to exhibit a wide variety of complicated phenomenon. In a simple case, a CA consists of a line of cells or sites, each with value of 0 or 1. These values are updated in sequence of discrete time steps, according to a definite fixed rule. Denoting the value of a site at position i in $(t+1)$th time step by a_i^{t+1}, a simple rule gives its next value as

$$a_i^{t+1} = F\left(a_{i-1}^t, a_i^t, a_{i+1}^t\right).$$

Here F is a Boolean function which specifies the rule, $a_{i-1}^t, a_i^t, a_{i+1}^t$ are the values of the cell itself and the immediately adjacent cells in the tth time step. This rule is called A-rule CA in this paper. Thus, there are altogether $2^8 = 256$ possible distinct CA rules in one-dimension CA with three variables. S. Wolfram [7] has studied the special properties of this rules and has discovered that there exists only 32 "legal" rules. Following [7], a rule is said to be "legal" if it is (i) reflection symmetric, and (ii) if given an initial state of all 0s, the state remains unchanged. Now, consider a class of CA with the following activation rule:

$$a_i^{t+1} = \Phi\left(a_{i-1}^t, a_i^{t-1}, a_{i+1}^t\right).$$

Here Φ is a Boolean function which specifies the characteristics of the rule. The value of a_i^{t+1} depends on the values of a_{i-1}^t, a_{i+1}^t (at the tth time step) and a_i^{t-1} (at the $(t$-1)th time step). This rule is called B-rule CA in this paper. Thus A-rule is equivalent to the 1st-order partial differential equations while B-rule the 2nd-order partial differential equations relating the input state variables. Likewise, we have discovered 32 "legal" rules for B-rule CA while the evolution of the patterns can further be partitioned to 5 classes as in Table 1. Moreover, only class 2 has the potential to be applied to image processing. In particular, rule B_{132} can be applied to binary image thinning. Fig. 1 shows some examples of the evolution of rule B_{132} with different initial conditions.

Binary Thinning with B-rule CA

Thinning has found a wide application in the field of character recognition in which, prior to encoding of a character, it is necessary to reduce the original image to simple lines [5]. Thinning is actually much more difficult than it looks, basically, two tasks must be implemented: (i) peeling the thick pixels off, and (ii) stopping the peeling process when the pixel size is exactly one or two. Our proposed algorithm B-rule CA can implement the above two tasks at the same time.

In this paper, we transform the image thinning process from a 2-D problem into two 1-D CA problems (vertical:Y and horizontal:X). In accordance with the evolution characteristics of B_{132} CA, any odd number of '1s present at the initial state leads to one-pixel thick stable state. Fig. 2 shows the schematic of the partitioned network. 2-D CA is separated into two 1-D CAs nominated with X-CA and Y-CA. To ensure the correct operations for the cells on the boundary, an "zero boundary condition" is assumed.

Fig. 3 shows some simulation results of a 40x40 pixels printed character. In Fig. 3a, the original character is presented, where the outputs by using Y-CA and X-CA are respectively, shown in Fig. 3b and 3c. Finally, combining the results at X and Y directions give Fig. 3d. Fig. 4 is the simulation results of a hand written Chinese character.

Implementation of Thinning Processor with CNN

B_{132} CA uses the following Boolean function:

$$a_i^{t+1} = \Phi\left(a_{i-1}^t, a_i^{t-1}, a_{i+1}^t\right) = a_i^{t-1} \cdot \left(a_{i-1}^t \oplus a_{i+1}^t \right),$$

where O is the Equivalence operator. The schematic of the above Boolean function implemented by logic gates is shown in Fig. 5. It is known that CNN can finish logical operations like AND, OR and NOT [8]. Thus, the above Boolean function can also be implemented by using CNN cloning templates. Fig. 6 shows the whole schematic of our proposed image thinning processor. The X-CA and Y-CA are basically time-discrete CNN [9].

To demonstrate the performance of our binary image thinning processor. Fig. 7 shows the simulation results of two hand written English characters "a" and "y" in a 40*x*40 pixels plane. In fact the size of a convex polygon in the X and Y directions can also be determined by our thinning processor. Fig. 8 shows some examples of determining the sizes of convex polygons.

By separating the 2-D thinning process with two 1-D thinning processor, the whole image processing is speed up. In general, the convergent steps are less than *n* for a *nxn* pixels plane. However, there are some minor disadvantages in our processor: (i) two layers of processing elements are needed, and (ii) the final thinning outcome may be discontinueu. However, the simplicity of implementation and higher processing speed makes it a vital component for image processing. Our research for the properties of B-rule CA and its application to image processing is only the first step. Another example of applying CA rules to the design of CNN for image processing like edge detection and connected component detector can also be found in [10]. The relations between CA and CNN and their potential capability of combining together for image processing worth further studying.

Conclusions

A class of new CA (B-rule) is ideal tool for the design of CNN for binary image processing (like image thinning). The potential capacity of our approach worth further studying.

References

[1] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory," *IEEE Trans. on CAS*, vol. 35, pp. 1257-1272, 1988.

[2] L. O. Chua and L. Yang, "Cellular Neural Networks: Applications," *IEEE Trans. on CAS*, vol. 35, pp. 1273-1290, 1988.

[3] L. O. Chua and B. Shi, "Exploiting Cellular Automata in the design of Cellular Neural Networks for binary image processing," *ERL memo UCB/ERL M89/130, University of California, Berkeley*, Nov. 15, 1989.

[4] T. Roska and L. O. Chua, "Cellular Neural Networks with Nonlinear and Delay-type Template Elements," *Proc. of IEEE Int. Workshop on Cellular Neural Networks and their Applications CNNA-90*, pp. 12-25, 1990.

[5] E. S. Deutsch, "Thinning algorithms on rectangular, hexagonal and triangular arrays," *Commun. ACM* 15, pp. 827-, 1972.

[6] T. Matsumoto, L.O. Chua, and T. Yokohama, "CNN Cloning Template: Image Thinning with a Cellular Neural Network," *IEEE Trans. on CAS*, vol. 37, no. 5, pp. 638-640, 1990.

[7] S. Wolfram, "Theory and Applications of Cellular Automata," *World Scientific*, 1986.

[8] T. Rosta, A. Radvanyi, T. Koxek, T. Boros, "Dual CNN Software," Report No. DNS-7-1991, Computer and Automation Institute of the Hungarian Academy of Sciences (MTA SzTAKI), Budapest, September 1991.

[9] H. Harrer and J. A. Nossek, "A Learning Algorithm for Time-Discrete Cellular Neural Networks," *IJCNN Singapore*'91, pp. 717-722, 1991.

[10 C. Ho, D. Yu, X, Yu and S. Mori, "Applications of Cellular Automata to the design of Cellular Neural Network for Image Processing," under preparation.

Class	B-rules
1	0, 32, 72, 127, 160
2	4, 36, 76, 104, 108, 132, 164, 200, 204, 232, 236
3	8, 22, 50, 54, 146
4	150,178,182,218,250,254
5	90, 94, 122, 126

Table 1

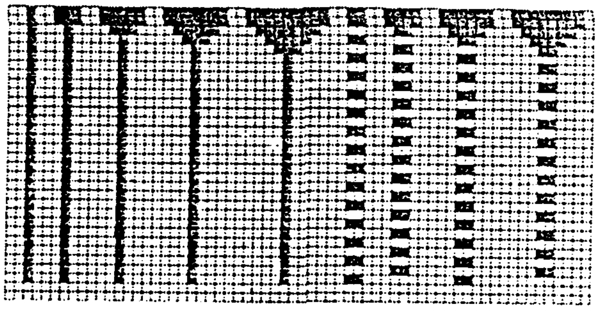
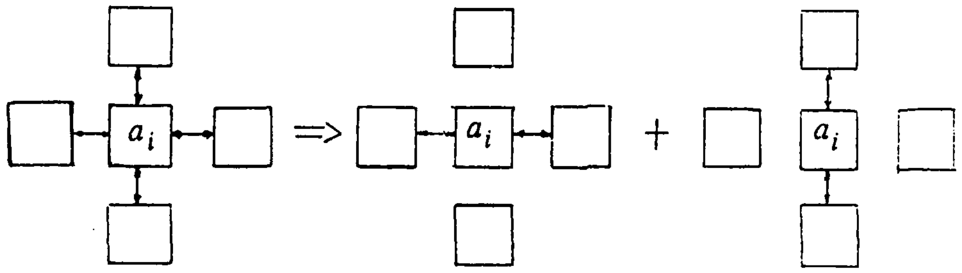
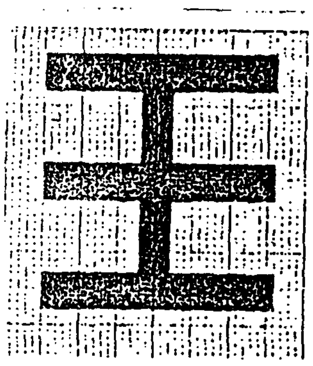


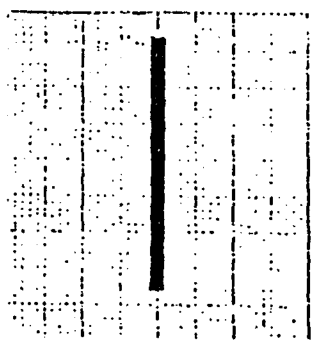
Fig. 1 Some evolution examples of rule B_{132} CA

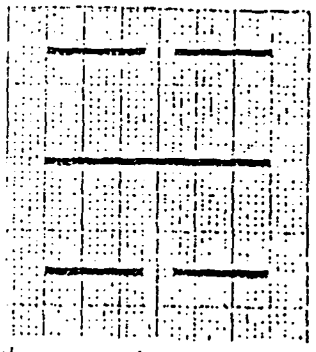


2DCA => X-1DCA + Y-1DCA

Fig. 2 Transformed from 2DCA into X-1DCA and Y-1DCA


(a) the original


(c) the output by using X-CA


(b) the output by using Y-CA

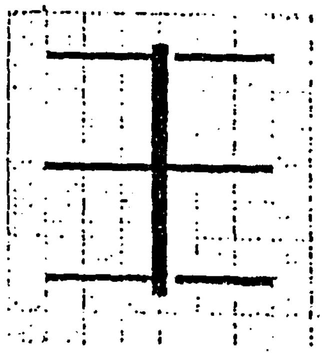

(d) combining results

Fig. 3 Simulation results of printed character

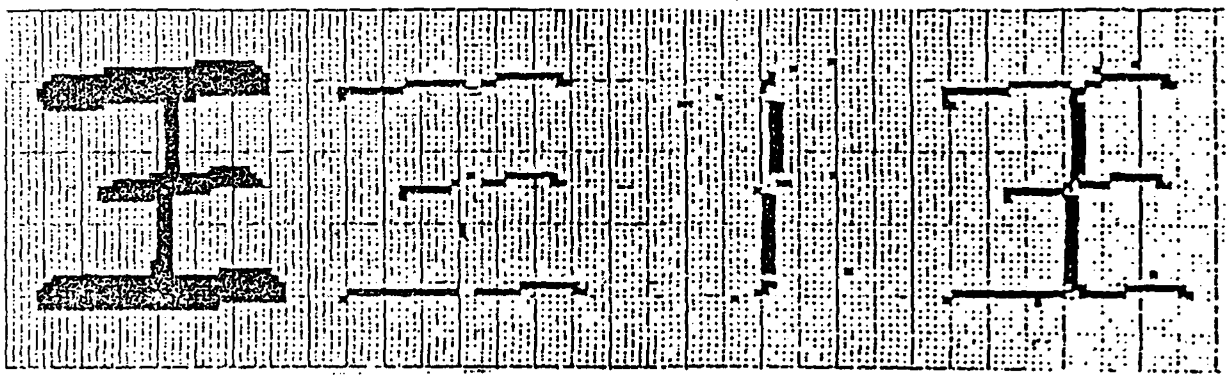
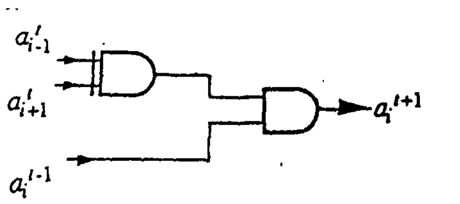


Fig. 4 Simulation result of a handwritten Chinese character



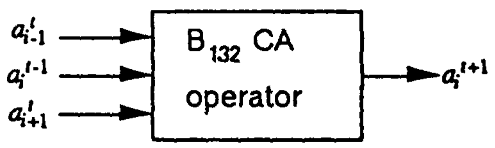


Fig. 5 B_{132} CA Operator

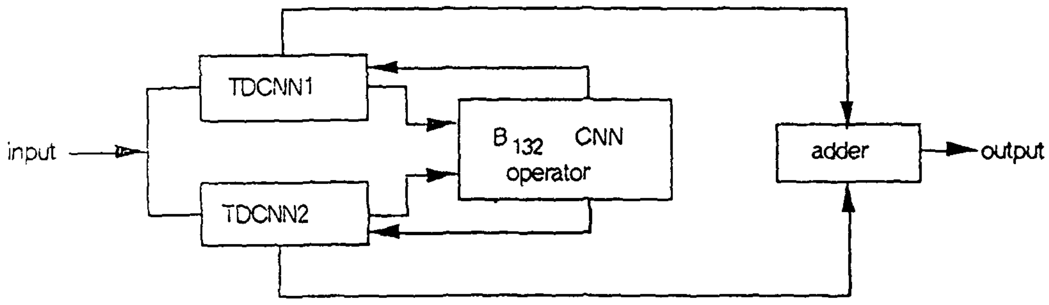


Fig. 6 Schematic of the thinning processor with CNN

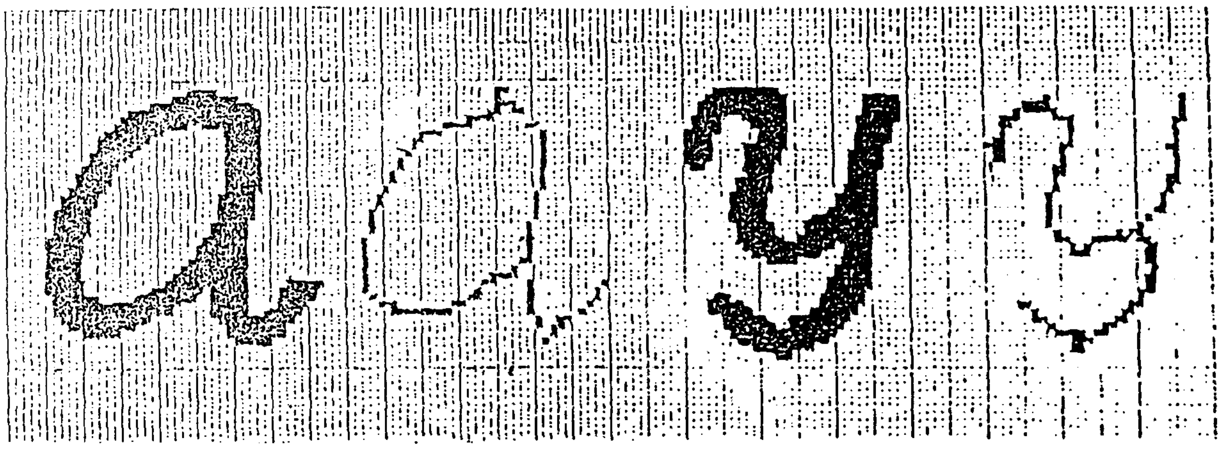


Fig. 7 Simulation results of two handwritten English characters by CNN processor

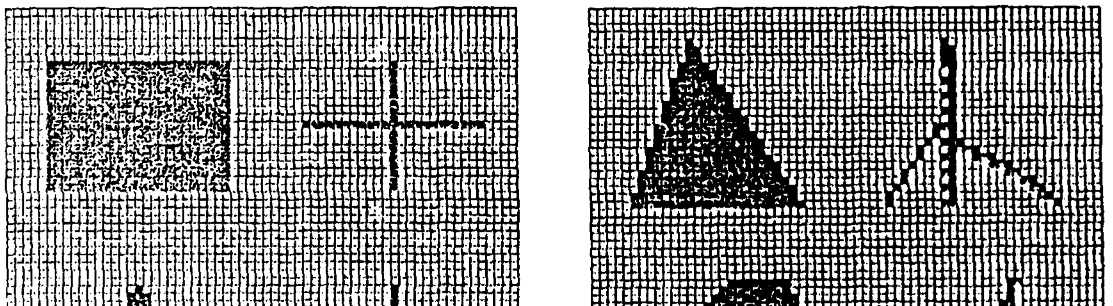


Fig. 8 Determining the sizes of convex polygons by CNN processor

Optical Tracking System for Automatic Guided Vehicles Using Cellular Neural Networks

Gy. Erőss, T. Boros+, Á.Kiss++, A. Radványi+, T. Roska+, J.Bitó, J.Vass++

Tungsram TH Co. Ltd, Centre of Robotics and Automation, Váci ut 77, Budapest, H-1340

+ Dual and Neural Computing Systems Laboratory, Computer and Automation Institute of the Hungarian Academy of Sciences, P.O.B.63, Budapest, H-1502

+ + Dep. of Information Technology, Faculty of Engineering, University Veszprém, Egyetem u. 10, Veszprém, H-8201

1. Abstract

Computer Integrated Manufacturing (CIM) Systems having determining role in the modern industry. These systems contain essentially two different equipments for transporting materials between workstations:

- conveyors can be used if the path of transportation is fixed and known in advance, or

- if the path of the transportation may change or there are various tasks to be solved along the path, the use of Automatic Guided Vehicles (AGV) is strongly recommended.

The path control of an AGV in CIM systems is generally solved either by following an inductive wire or by determining its position and orientation from signals provided by transmitters mounted at some characteristic points of the workshop. Both of these basic methods have their advantages and drawbacks.

In this paper a new method will be proposed for path-control, which combine the flexibility and easy installation of optical methods with simplicity and robustness of the inductive method. Using a new computing paradigm, the Cellular Neural Network (CNN) [1],[2] and a related device [3], the VLSI CNN chip, a very high speed solution can be achieved, that is less expensive than the conventional methods - keeping their advantages. This AGV control comply with the requirements of CIM systems. Further advantages of the proposed system are as follows: fault tolerance and ability to give instructions along the path, and the use of a simple local control.

2. The proposed new method

In this report the possibility and feasibility of using CNNs for control of AGV is investigated. The schematics of the control unit can be seen in Fig.1. The input sensory array may work either in the visible electromagnetic range or, eventually, in the ultraviolet or infrared spectral range. In this application, the path determined by series of control patterns and the AGV has to trace these, and deduce the motion speed and direction. The input picture obtained is directly forwarded to the CNN unit. The CNN unit has to solve the pre-processing (filtering, noise-reduction, pattern restoration, etc.) of the detected picture and the analysis of the shape, size, and orientation of control patterns. A simple logic makes the desired control signal from the result of analysis. This method is new in control, because CNN was never used in this field. The results of using CNN for character and pattern recognition are applied combining with optical feedback.

The main advantage of this method:

- using analog VLSI and integrated device (see Part 5) a real time control available;
- it's more flexible and less expensive than conventional methods;
- easy installation and variability;
- fault tolerance due to robustness of CNN and optical detection;
- improvements can be made easily due to the wide application of CNN (see Part 5).

The control hardware is less complex compared to conventional methods.

3. Determination and analysis of control pattern

For the proposed method a control pattern had to be designed, which meets the folloving conditions:

- it doesn't contain small parts, it is strongly fault tolerant and should contrast with the background;
- the desired direction can be determined from the induvidual pattern;
- it is symmetric so that the path can be followed there and back.

Keeping in mind these conditions, the elementary control pattern has been choosen as depicted in Fig.2., and its horizontal projection represents the desired direction. This pattern is contiguous therefore it can be filtered easily. The horizontal projection can be calculated by the well known CCD function [9]. The pattern has three main parameters : a, β, and γ. They define the figure unambiguously. The absolute size of the pattern (parameter a) can be set according to the size of the visual field of the detector. The horizontal projection of the pattern is determined by the length of its short and long diameter when the pattern is rotated in the sensory field. The tangent of the desired path direction is represented by the position, when the two diameter's horizontal projections are equal, so the AGV has to turn on the right when the projection is shorter, and on the left when it is longer.

The length of the projection :

$$h(\varphi) = \max\{x \sin(\varphi+\beta), y \cdot \sin(\varphi+\gamma)\}$$

where φ is the rotation acording to the direction to be followed, and it can be in the intervall $[-\alpha, 90^\circ-\beta]$ - α can be computed from the parameters β and γ. Using the pattern described above and it's horizontal projection, the inverse of the last function can be used to determine the orientation with respect to the tangent of the desired path (Fig.3.).

4. Experiments

To investigate the proposed method three different experiments have been carried out in order.

4.1 Experiment 1.

The goal of this experiment is to prove that the control pattern can be detected and processed in noisy background, and verify that the information can be decoded from the detected picture. According to the principles stated above as well as to practical considerations, the main parameters of the control pattern have been chosen to:

$$a=5.25 \text{ cm}, \quad \beta=41.88^\circ, \qquad \gamma=57.11^\circ.$$

A series of control patterns has been scanned by means of a camera. An individual control pattern has been cut out from the stream and digitized with a frame grabber (Fig.4.). The noisy image of resolution 44x44 has been fed into a CNN program package [8]. The simulator computed and displayed the transient of the CNN at discrete time-intervalls. After the transient has settled down, the steady state response of the CNN remains on the screen. In reality, using analog VLSI realization of CNN, this transient settles down in a few microseconds. For filtering and CCD function the following well known templates were used consecutively [1.b][9]:

filtering: $$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}.25 & .25 & .25\\ .25 & 2 & .25\\ .25 & .25 & .25\end{bmatrix} \qquad I=2;$$

CCD: $$A=\begin{bmatrix}0 & 0 & 0\\ 1 & 2 & -1\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad I=0;$$

The simulation results clearly shows that the control pattern can be succesfully filtered and restored from a noisy background (Fig.5.). The projection length computed by the CNN simulator and displayed at the right side of the output picture (Fig.6.) is proportional to the angle of rotation needed to follow the path.

4.2 Experiment 2.

The main goal of this experiment was to investigate whether the method can be applied for controlling a real robot or not. The schematics of the experimental arrangement can be seen in Fig.7. The camera scan a control pattern representing a part of the path to be followed. The picture is fed into a digitizer and the digitized picture is processed by an IBM-AT compatible computer. A special add-on-board [12] is used for the simulation of a CNN - it computes the result faster than the software simulator applied in experiment 1. The result of which is the angle calculated from the horizontal projection of the scanned pattern is fed into the control of a SCARA type robot, and the gripper of the robot moving along an arch required, and represented by the scanned pattern. If the camera mounted directly at the gripper, this experimental arrangement can be used for teaching-in as well.

4.3 Experiment 3.

A software simulator program has been developed to test the proposed AGV control. On the screen the simulator shows on the screen the patterns indicating the desired path, the simulated image seen by a camera attached to the vehicle, yand the path wandered by the AGV. Using this simulator the effect of different parameters (e.g. acceleration of the vehicle, fine tracking, etc.) can be tested along any path given.

5. **Application, further improvements**

The VLSI and optical realization of CNNs are under extensive research [7][10][11]. The advantage of the presented method is that all the necessary electronic units - exept for power electric parts - can be integrated into one single VLSI chip. The chip would consists of two CNN layers performing image restoration (filtering, etc.) and pattern recognition and interpretation (CCD). To facilitate additional functions described below a CNN layer with programable templates is recommended. Integrating an additional sensory array in CNNs is already the subject of major research. Further simple logical control units may be attached to the design as well, to obtain a dual-computing intelligent sensory array.

One additional improvement to the method may be the 'fine tracking'. In the proposed method significant rotational deviations ($\varphi >$ apx. 1° - depending on resolution of sensory array and the CNN) can be corrected easily. However, especially along a long straight path, the necessity of fine tracking may be demanded. A signal for fine tracking can be obtained from the analysis of the bottom line of the CNN image after filtering: any little part of a control pattern detected in the "n" pixel-wide left/right margin would generate an appropriate control signal.

Besides the proposed method and device with additional CNN functions - e.g. character recognition - can be applied for interpreting other patterns placed among path-control patterns. In this way the AGV or the robot might recive instructions to perform various tasks like "stop for 'n' seconds", "do something", "act according to the environment", "determine the control strategy", etc. For example, the control pattern may be combined with vertical stripes, that don't influence the orientation and the horizontal projection of the pattern. The number of stripes may encode different instructions, and can be interpreted using vertical CCD. Further investigations are needed concerning the fault tolerance and robustness of such "enhanced" patterns.

6. **Acknowledgement**

The authors would like to thank Péter Lantos for making the program used in Experiment 3., and his helpful ideas.

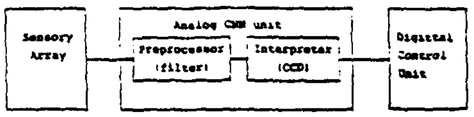


Figure 1 Sematic of control unit

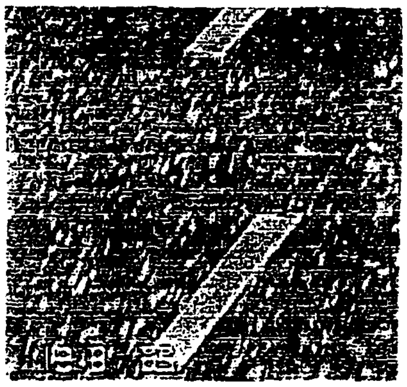


Figure 4 Digitized noizy input frame

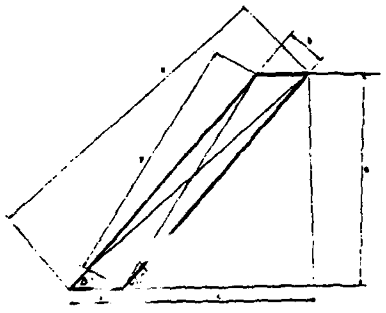


Figure 2 Elementary control pattern

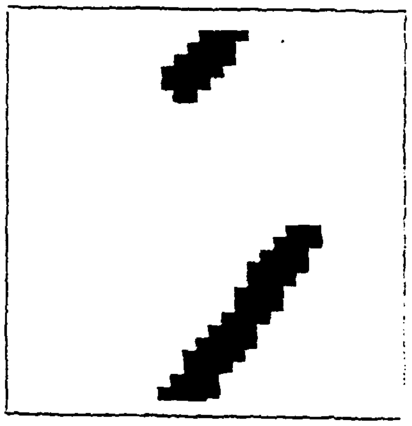


Figure 5 The filtered pattern

Figure 3 Desired path represented by control patterns

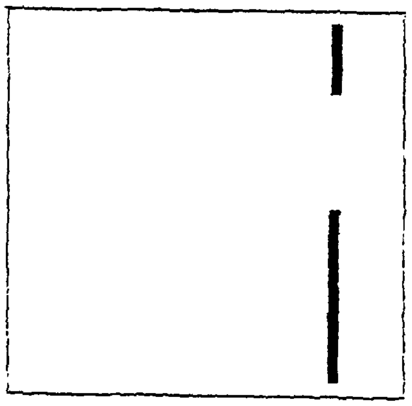
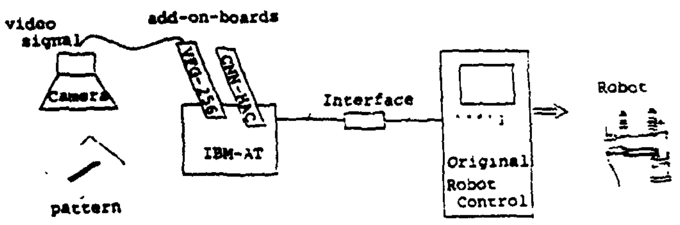


Figure 6 The output picture after CCD function

Figure 7 Experimental

arrangament (Exp.2.)



References

[1.a] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory", *IEEE Transactions on Circuits and Systems*, Vol.35, pp.1257-1272, October 1988.

[1.b] L. O. Chua and L. Yang, "Cellular Neural Networks: Application", *IEEE Transactions on Circuits and Systems*, Vol.35, pp.1273-1290, October 1988.

[2] T. Roska and L. O. Chua, "Cellular neural networks with nonlinear and delay-type elements", *Proceedings of the International Workshop on Cellular Neural Networks and their Applications CNNA'90*, IEEE Catalog No. 90TH0312-9, Library of Congress Catalog No. 90-81231, pp.12-25, 1990.

[3] L. O. Chua and T. Roska,"Stability of a class of nonreciprocal cellular neural networks", IEEE *Transactions on Circuits and Systems*, Vol.37, pp.1520-1527, December 1990.

[4] L.O.Chua and Chai Wah Wu, "On the Universe of Stable Cellular Neural Networks", *Memorandum* No. UCB/ERL M91/31, 24 April 1991

[5] T. Sziranyi, T.Roska and P. Szolgay, "A Cellular Neural Network Embedded in a Dual Computing Structure (CNND) and its Use for Character Recognition", *Proceedings of the International Workshop on Cellular Neural Networks and their Applications CNNA'90*, IEEE Catalog No. 90TH0312-9, Library of Congress Catalog No. 90-81231, pp.92-99, 1990.

[6] T. Roska, T. Boros, P. Thiran and L. O. Chua, "Detecting simple motion using cellular neural networks", *Proceedings of the International Workshop on Cellular Neural Networks and their Applications CNNA'90*, IEEE Catalog No. 90TH0312-9, Library of Congress Catalog No. 90-81231, pp.127-138, 1990.

[7] K. Halonen, V. Porra, T. Roska and L. O. Chua, "VLSI implementation of a reconfigurable cellular neural network containing local logic (CNNL)", *Proceedings of the International Workshop on Cellular Neural Networks and their Applications CNNA'90*, IEEE Catalog No. 90TH0312-9, Library of Congress Catalog No. 90-81231, pp.206-215, 1990.

[8] *CNND simulator - Cellular Neural Network embedded in a simple Dual computing structure, User's guide Version 3.01*, Report No.37/1990, Computer and Automation Institute of the Hungarian Academy of Sciences (MTA SzTAKI), Budapest, January 1990.

[9] T. Matsumoto, L. O. Chua and H. Suzuki, "CNN Cloning Template: Connected Component Detector", *IEEE Transactions on Circuits and Systems*, Vol.37, 633-635, May 1990.

[10] N.Frühauf and E.Lüder, "Realization of CNNs by Optical Parallel Processing with Spatial Light Valves", *Proceedings of the International Workshop on Cellular Neural Networks and their Applications CNNA'90*, IEEE Catalog No. 90TH0312-9, Library of Congress Catalog No. 90-81231, pp.281-290, 1990.

[11] C. Mead and M. Ismail (Eds.), *Analog VLSI implementation of neural systems*. Boston: Kluwer, 1989.

[12] T. Roska, G. Bártfai, P. Szolgay, T. Szirányi, A. Radványi and Zs. Ugray, "A hardware accelerator board for cellular neural networks:CNN-HAC", *Proccedings of the International Workshop on Cellular Neural Networks and their Applications CNNA'90*, IEEE Catalog No. 90TH0312-9, Library of Congress Catalog No. 90-81231, pp.160-168, 1990.

[13] T. Roska, "Analog events and a dual computing structure using analog and digital circuits and operators", in *Discrete Event Systems: Models and Applications*, P. Varaiya and A. B. Kurzhanski (Eds.), Berlin: Springer-Verlag, 1987.

[14] J. Cruz and L.O. Chua, "A CNN chip for connected component detection", IEEE Trans. Circuits and Systems, Vol. 38, pp. 811-812., 1991.

A DUAL CNN MODEL OF CYCLOPEAN PERCEPTION AND ITS APPLICATION POTENTIALS IN ARTIFICIAL STEREOPSIS

A.G.Radványi
Dual and Neural Computing Systems Laboratory
Computer & Automation Institute (MTA SzTAKI) Research Division, Hungarian Academy of Sciences
Kende u. 13, H-1111 Budapest XI, Hungary; H-1518 Bp, POB 63; E-mail: h189rad@ella.hu

Abstract

The random-dot stereogram coding 3D information in its internal correlation is for probing human stereopsis. We report dual CNN algorithms that can reveal 3D surfaces coded in stereograms. The concept of difference stereogram is introduced and used for coding smooth surfaces. Its importance is due to the fact that difference stereograms of real objects can be created in an optical environment using projector and camera.

INTRODUCTION

An interesting question about the human visual system is whether we identify an object before we put it in its proper perspective, or whether there exists a mechanism - the Cyclopean eye - in the visual cortex, that can perceive "pure" depth based merely on the correlation and parallax of the left and right retinal images. Using random-dot stereograms [4] almost anyone can try and testify to oneself the existence of the Cyclopean mechanism.

The random-dot stereogram (RDS) is a two-dimensional, seemingly random pattern consisting of pairwise horizontally correlating internal segments. Shifting a RDS horizontally upon itself, correlating (identical) areas will overlap again and again. Viewing properly an RDS devoid of any (monocularly) observable cue, the Cyclopean eye will find the correlating areas and "see" them in different perspective depths, depending on their horizontal distance on the stereogram.

In Section 1, methods of creating different types of RDS's - stereo pairs, auto-stereograms and difference stereograms - are discussed, laying stress upon the internal structure of stereograms.

In Sections 2 and 3, dual CNN algorithms that can find the perspective hidden in stereograms, are dealt with.

In Section 4, an optical set-up with projector and camera, to produce difference stereograms of real objects is outlined. Potentially, if connected to a dual CNN hardware it can detect 3D depth variations in its scope.

1. DIFFERENT TYPES OF RANDOM-DOT STEREOGRAMS

The RDS is a visually perceptible rectangular pattern printed on paper or displayed on a screen, conveying 3D depth information coded in its internal correlation. Its mathematical representation, which for simplicity we also call RDS, is a two-variable function $g(x,y)$ coding a depth pattern represented by the $s(u,v)$ surface function. In visualization the $g(x,y)$ values are converted into different visual attributes as colour, brightness or texture.

In RDS's, depth is transformed into correlation, which is defined as follows:

g at (x_c,y) and (x_c+d,y) is fully correlated, if $g(x,y)=g(x+d,y)$ for $x_c \le x \le x_c+l$, where l is a suitable correlation length. (For partial correlation the equality need not hold uniformly over the total correlation length.)

On its either - say left - side, the RDS contains a rectangular area with no correlation inside. Almost all the rest of RDS is composed of patches that are copies of some other areas to the left of them. The copy mechanism ensures internal correlation in the above sense. The position of areas to be copied is determined in accordance with the depth pattern to be coded.

The different types of RDSs are defined by the following formulae:

Let x_D and y_D respectively be the horizontal and vertical dimensions of the RDS, and $r(x,y)$ an internally uncorrelated pattern for $0 \leq x < x_D$ and $0 \leq y < y_D$; and let x_P be the width of the left uncorrelated area. We define the $S(u.v)$ partial slope function of the surface in the horizontal direction as the limit of $(s(u+D,v)-s(u,v))/D$ with D approching 0. We also introduce a $c(s)$ copy-source-selector function to be defined later, according to the different types of RDSs.

Then for $0 \leq y < y_D$,

$$
\begin{aligned}
g(x,y) &= r(x,y) && \text{if } x < x_P, \\
g(x,y) &= g(c(s(x-x_P,y)),y) && \text{if } x \geq x_P \text{ and } S(x-x_P,y) < 1, \\
g(x,y) &= r(x,y) && \text{if } x \geq x_P \text{ and } S(x-x_P,y) \geq 1.
\end{aligned}
$$

All the quantities above, i.e. coordinates, RDS and depth values can be either real or integer values. The width x_P sets constraints upon the dynamics of depth variations for the sake of a meaningful resulting stereogram.

If $x_D = 2 * x_P$, we obtain a random-dot stereopair, with either half uncorrelated in itself. For visual inspection its halves can be separated and projected onto the left and right retinae.

If x_D is some multiple of x_P, we obtain an auto-stereogram [5], that can be inspected without any special device, simply fixating behind or before the plane of the RDS. The two fixation positions give reverse depth perception with respect to each other. Fig. 2 shows the auto-stereogram of the step-pyramid seen in Fig. 1.

For stereopairs and auto-stereograms the copy-source-selector function is the following: $c(s) = x - x_P + s$.

Using $c(s) = (x - x_P + s) \bmod x_P$ as the copy-source-selector function leads to difference stereograms. The difference stereogram is for coding the changes in surface depth, instead of the depth itself. It has two distinct advantages from the aspect of depth coding. One is the reduced requirement for width x_P due to the fact that changes of depth on surfaces are usually smaller than the absolute depth. The other is that difference stereograms can be produced by simple optical devices too, as will be seen later.

2. DUAL CNN ALGORITHM TO EXTRACT DEPTH FROM RDS'S

When perceiving depth, the binocular fusion mechanism of the human visual system picks out fields correlating in the two retinal images. Correlation is tested by overlapping the retinal images at different relative shifts. Based on an entropy-like measure - called neurontropy [6] - maximum correlating areas together with their relative shifts which are directly proportional to the depth of corresponding areas on the coded surface, are extracted.

The function of this neural mechanism can be broken into steps realizable in the framework of the Dual CNN paradigm [2], which contains logic operations in addition to the analog CNN ones. Binary (say black on white) random-dot stereograms coding surfaces that are discretized in all three dimensions, can be processed in the dual CNN framework.

The dual CNN algorithm can find the surface steps in depth, one after the other, in successive phases. Each phase results in the image of surface segments, the fragments of the whole surface that lie in the depth layer just investigated. Since in the dual CNN framework [3] an external control is supposed to exist, the value of depth just investigated is always known. Combining the results of successive phases, yields also the level line structure of the whole surface.

In the following outline of the algorithm, the RDS is split into left and right images. Both images and the results as well, are x_D-x_P wide, the left image being the left part of the RDS and the right one its right part of the same width.

The N-th phase of the algorithm consists of the following steps:

Step 1.: We shift the right image by one pixel to the left and replace it with the result. Depending on the dual CNN device used shifting can be performed in two different ways. If the right image can always be stored inside the dual CNN - e.g. in its logic register, then a simple template, having only one non-zero entry, $B_{23}=1$, will do the shift. However, if the right image is stored in external memory, loading it into the CNN from gradually increasing start addresses will substitute for the shift.

Step 2.: The task to find in the left and right image, sufficiently large correlating areas having the same pattern, involves the following steps.

Step 2/a.: We perform a logical *equivalence* operation that by making pixel level correlations, will immediately reveal to an onlooker, the areas being searched for. The image obtained is contiguously black on correlating areas and shows random black and white noise pattern elsewhere.

Step 2/b.: To demarcate correlating areas from noise by CNN, we introduce the next simple criterion: the black pixels having no white neighbours (most probably) belong to correlating areas. Using the *edge* [7] template, that inverts the internal pixels in black areas, we can find just those pixels that fulfil the criterion.

Step 2/c.: The final step in finding correlating areas is the removal of the noisy pattern. Combining by logical *or* operation, the result of *edge* (Step 2/b) and the inverted result of logical *equivalence* (Step 2/a) will yield the white image of the N-th layer of the coded surface, on a black background.

The logical *inversion* and logical *or* operations can be performed consecutively by the logic part of dual CNN. Equivalently, a composite *inverse-or* operation can be defined and performed using the $(A_{22}=1, B_{22}=-1, I=1)$ analog template.

The success of this noise removal step is highly dependent on the local randomness of RDS, or - conversely - on the adequacy of the demarcation criterion above. According to experience, small error patches may remain in the results. Applving a small positive value in the feedback off-center positions of the *inverse-or* template can improve noise removal by filtering out isolated errors or even one pixel wide stubs, depending on the value applied.

Step 3.: Starting in the first phase with an "empty" black image, the surface layers found in succession can be aggregated by logical *and* operation to produce the level line structure of the surface coded. Essentially, the succession of logical *and*s cuts and pastes those edges that were found in the demarcation step (2/b). In addition, it has a noise filtering side effect in the sense that any errors that lie entirely inside areas on other depth layers (before or behind in depth) will disappear. On the other hand, errors that cross surface edges will locally corrupt the level line result. Fig. 3 shows the level lines of the step-pyramid seen in Fig. 1.

3. THE DIFFERENCE RDS AND ITS USE WITH REGULAR DOT PATTERN

Producing a difference RDS effectively results in the storage of the difference between the depth of each fragment and another fragment, a distance x_p to the left. In other words, the difference RDS of a surface is the auto-stereogram of the "difference" surface, that can be produced simply if we divide the original surface into x_p wide vertical bands and - keeping the leftmost band unchanged - reduce the depth of each fragment by the depth of the corresponding fragment in the adjacent band to the left. Using a small enough x_p with respect to the minimum horizontal distance of unit depth changes, yields a three-level $(-1, 0, +1)$ difference surface. The surface of only 2 bit dynamics can be coded with small x_p and vice versa, that allows fine spatial resolution. Fig. 4-a and Fig. 4-b show the difference surfaces of the 76 pixel wide step-pyramid, for $x_p = 2$ and $x_p = 12$, respectively.

From the aspect of processing RDS's with a CNN, the difference RDS, due to its narrow correlation band, is fundamentally different from the other types of RDS's. The global correlation property of those is transformed into a local one of difference RDS's, that can be detected in a simple CNN window of neighbourhood 2 or 3. By that way, both flat areas and ascending or descending slopes of the coded surface, can be extracted in respective, single CNN steps directly from the RDS.

In case of narrow band difference RDSs the randomness of the pattern looses its importance. Just the knowledge of pattern structure may contribute to the reconstruction of the coded surface. Let us consider a discretized surface that is "smooth" with respect to the pattern density. (Smoothness means, that the depth difference of neighbouring pixels are at most 1, and none of the 2 by 2 pixel areas form a saddle.) Let us produce a narrow band difference stereogram of $x_p = 4$, using a regular checker pattern of squares of 2 pixel long edges (Fig. 5). Smooth surfaces coded in checker pattern can be fully reconstructed by a dual CNN algorithm with 22 analog templates of neighbourhood 1, in 3*3 window, as follows:

Step 1: Find vertical edges where the surface is ascending from left to right

Step 2: Find vertical edges where the surface is descending from left to right

Step 3: Find all the horizontal edges

Step 4: Using the "smooth surface" property and the already known sort of vertical edges, identify the downward descending and downward ascending horizontal edges, respectively.

Result 1: The union of edges found gives the level lines of the surface (Fig. 6)

Step 5: Propagating the shadow of ascending/descending edges until it reaches a descending/ascending edge gives the ascending/descending slopes both in horizontal and vertical direction.

Step 6: The intersection of ascending and descending slopes gives the row-wise/column-wise local maxima/minima, in other words the vertical/horizontal ridges/valleys of the surface.

Step 7: Calculate the intersection of vertical and horizontal ridges/valleys. Erase those parts of the result which are not bordered by some closed level line loop exactly.

Result 2: Result of Step 7 yields the local maxima/minima (peaks/hollows) of the surface.

Result 3: Using modified CCD to propagate ascending/descending edges until reaching a descending/ascending edge, the height/depth of local maxima/minima can be measured, from the direction of propagation (Fig. 7).

4. THE CREATION OF DIFFERENCE RDS WITH OPTICAL MEANS

Aside from its inevitable importance in vision research, up until this point to produce and probe RDS may have seemed an enjoyable and aesthetic play, devoid of any significance from the engineering point of view. However, in principle, diffeıence RDS's of real objects can be produced directly by optical means. Moreover, research of optical CNN devices is reportedly under way [8], that together with the possibility of making continuous, real time difference RDSs of surrounding objects may lead to a "stereoscopic" robot eye.

Let us consider a projector and a camera side by side at the same height in front of a white wall. Let us project perpendicularly onto the wall a horizontally periodic random-dot (or checker) pattern, i.e. the RDS of a single plane, from such a distance that the parallax over the projection area be negligible. The camera - from beside the projector - will see the whole area at a specific angle, that depends on its distance from the projector, and is the same for the whole area. In fact, the camera will see the RDS, or what is the same in this case, the difference RDS of a single plane.

Let us place some angular objects in front of the wall in such a way, that their surfaces be either parallel with or perpendicular to the wall. From the projectc.: no change in the projected pattern can be noticed. However, from the camera, definite horizontal shifts of pattern segments that now fall on the inserted objects, can be detected. The resulting view from the camera is the difference RDS of the whole arrangement, except for the images of those surfaces, which are perpendicular to the wall. They go "empty" in the optical method and would be filled with an uncorrelated pattern in the computer algorithm. A camera image of objects in Fig. 8-a is shown in Fig. 8-b. The amount of horizontal shift is dete.mined by the change in depth and the camera angle. We consider depth resolution to be that change in depth, which causes one pixel horizontal shift in the camera image. For a given pixel size, the depth resolution can be adjusted by the camera angle.

Presumably, using electronically controlled devices, the surface in front of the above set-up, can be scanned row-wise by a flying window and processed in real time by a tiny CNN array. Moreover, in a feedback loop, the resolution and the size of the flying window can be adjusted adaptively to the extracted local features of the surface.

REFERENCES

[1] a L.O.Chua and L.Yang, "Cellular neural networks:Theory", IEEE Transactions on Circuits and Systems, Vol.35, pp.1257-1272, 1988.

b L.O.Chua and L.Yang, "Cellular neural networks: Applications", ibid., pp.1273-1290.

[2] T.Roska and L.O.Chua, "Dual CNN analog software", Report DNS-1-92, Dual and Neural Computing Systems Res.Lab., Comp. Aut. Inst., Hung.Acad.Sci., 1992.

[3] a K.Halonen, V.Porra, T.Roska and L.O.Chua, "VLSI Implementation of a reconfigurable CNN containing local logic", Proc. CNNA-90, pp.206-215, 1990

b A.Radványi, K.Halonen and T.Roska, "The CNNL Simulator and some time-varying CNN templates", Report DNS-9-91, Dual and Neural Computing Systems Res.Lab., Comp. Aut. Inst., Hung.Acad.Sci., 1991.

[4] a B.Julesz, "Binocular Depth Perception of Computer-Generated Patterns", Bell Syst. Tech. J. 39, pp. 1125-1162, 1960.

b B.Julesz, "Foundations of Cyclopean Perception", Chicago University Press, Chicago, 1971.

[5] C.W.Tyler, "Sensory processing of binocular disparity", in C.M.Schor & K.J.Cuiffreda (Eds.),"Vergence Eye Movements", pp. 199-295, Boston; Butterworth, 1983.

[6] B.Julesz, C.W.Tyler, "Neurontropy, an Entropy-Like Measure of Neural Correlation, in Binocular Fusion and Rivalry", Biological Cybernetics 23, pp. 25-32, 1976.

[7] T.Roska, A.Radványi, T.Kozek and T.Boros, "Dual CNN software library", Report DNS-7-1991, Dual and Neural Computing Systems Res.Lab., Comp. Aut. Inst., Hung.Acad.Sci., 1991.

[8] N.Frühauf and E.Lüder, "Realization of CNNs by optical parallel processing with spatial light valves", Proc. IEEE CNNA-90, pp. 281-290, 1990.

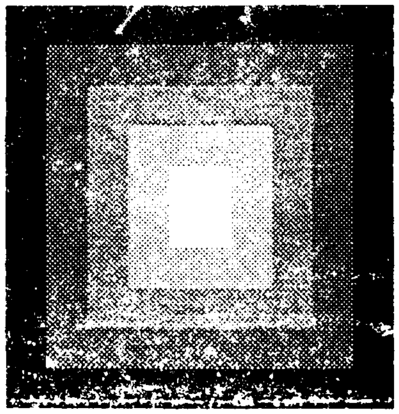


Fig. 1.

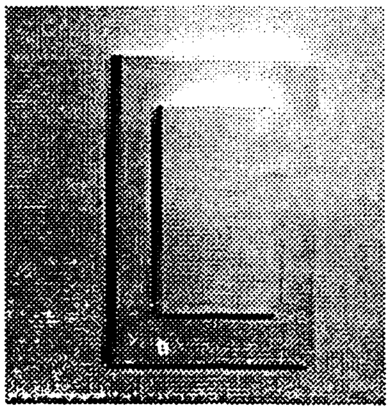


Fig. 8-a.

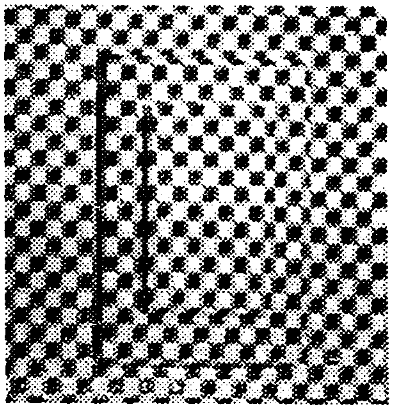


Fig. 8-b.

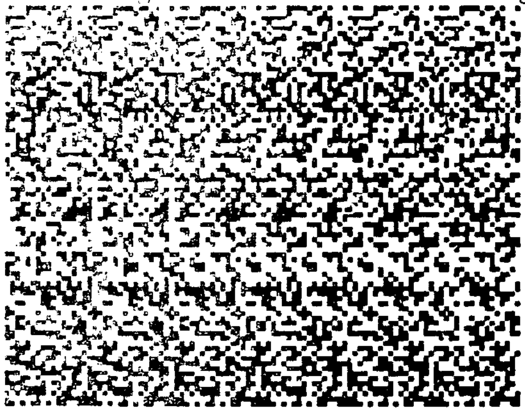


Fig. 2.

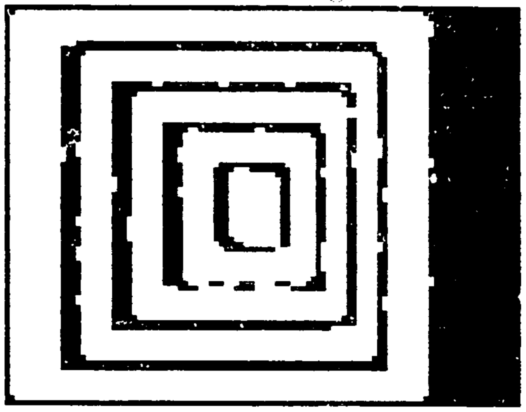


Fig. 3.

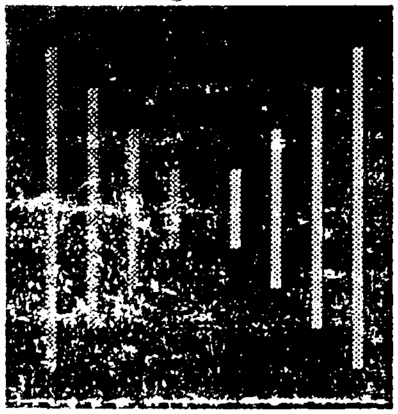


Fig. 4-a.

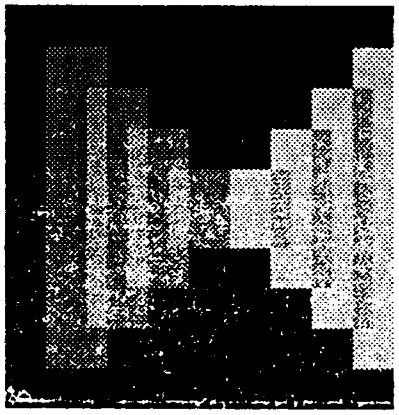


Fig. 4-b.

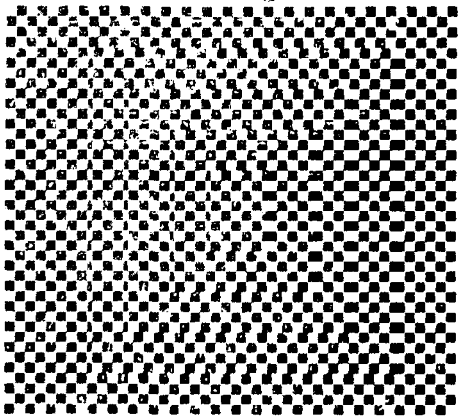


Fig. 5.

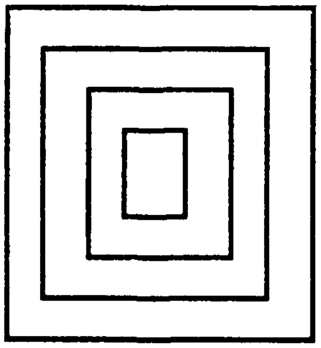


Fig. 6.

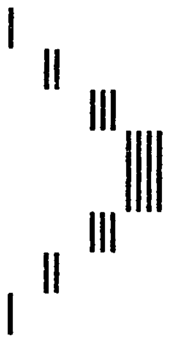


Fig. 7.

A Flexible PC-Controlled Analog/Digital Test Set for CNNs

by Immanuel Köhn, Bernhard Smith, Hubert Harrer and J. A. Nossek

Technical University Munich, Inst. f. Network Theory and Circuit Design
Arcisstr. 21, D-8000 Munich 2, Germany

Abstract

This paper describes a flexible test set that allows PC-controlled analog and digital measurements of a test object. The concept is modularly structured and enables an individual constellation depending on the testing object such as a CNN chip. Analog and digital signals can be transferred to and from the object. The communication to the PC is realized via a standard interface. For the PC, a test program language (TPL) has been developed, which is easy to handle and allows a flexible and completely automatic testing of a chip. The results of the measurement (e.g., dc characteristics) can be graphically visualized. For dynamic measurements the bottle neck consists in the data transfer to the PC and the processing of the testprogram interpreter. To overcome this problem special hardware modules have been developed, which allow a cyclic or linear output of analog and digital signals up to 10 MHz. Their core consists of a FIFO, which is preloaded before the processing. The output rate is controlled by a clock and different modules are synchronized via trigger signals.

1 Introduction

As commercial chip testers are very expensive, the necessity to develop a economic test set for the test of the first DTCNN chip [1] was given. The idea was to built up a system that is not fixed to a special structure of the test object [2]. It was designed with respect to

- a simple measurement by PC with a minimal number of additional wire connections
- a flexible and expandable structure in hardware and software
- an automatic DC characteristic measurement.
- a dynamic testing up to 10MHz.
- a software with an interface to C-routines.

This has been achieved by a modular structure, where the test set consists of a number of single modules, which communicate with the PC via a standard periphery bus by the I/O card [3], [4]. Each module has an identification address and a status register. Thus the operation of multiple modules of the same type is possible. They are connected by a port or a BNC connector to the test object (e.g., a CNN chip), which is plugged on a connection card. The advantage of this concept is that only the connection card has to be rebuilt for a new chip. The required modules

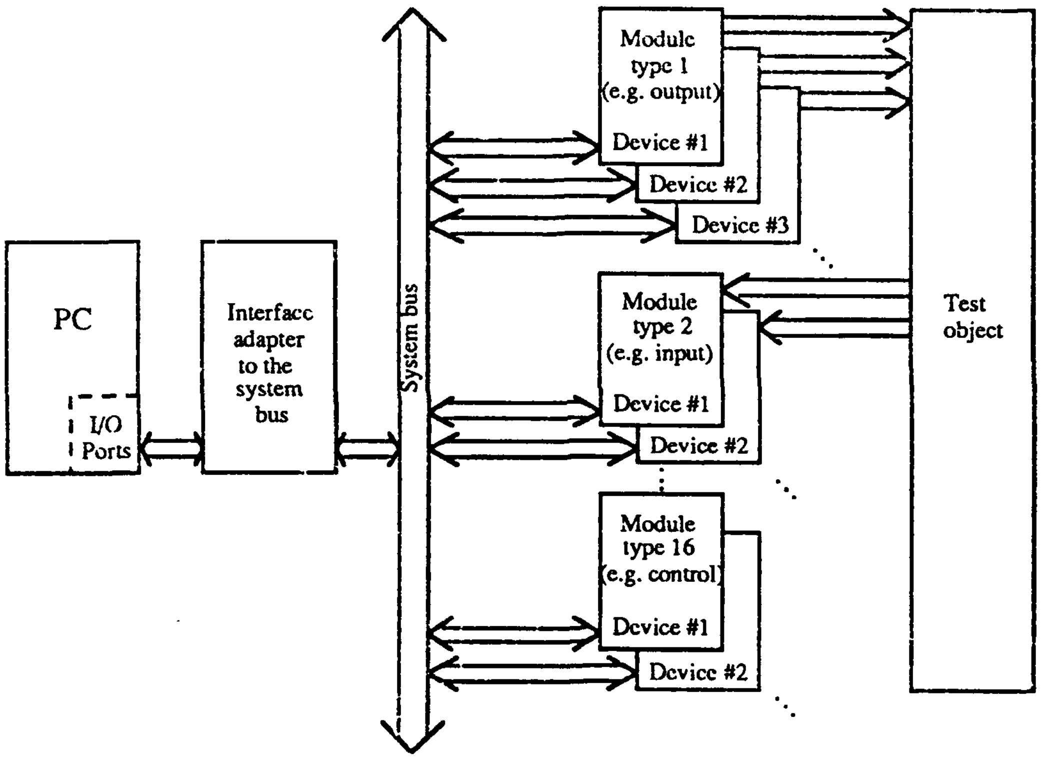
0-7803-0875-1/92 3$.00 ©1992IEEE

are simply plugged on the ports of a bus card. For additional requirements, new modules can be developed, if the same bus protocol is used.

Closely related to the flexible hardware structure, the software has to support this concept in flexible and expandable routines. It is based on the assignment of logical signal names to hardware addresses. A hardware configuration list allows the user to operate only with logical signal names. Each signal is uniquely assigned to a channel on a specified module So, a configuration table has to be created at first for each test object, which also fixes the required types and numbers of modules.

2 Hardware Structure

2.1 Hardware Concept



PC
I/O Ports
Interface adapter to the system bus
System bus
Module type 1 (e.g. output)
Device #1
Device #2
Device #3
Module type 2 (e.g. input)
Device #1
Device #2
Module type 16 (e.g. control)
Device #1
Device #2
Test object

Figure 1: Hardware concept of the PC-controlled testing device.

The measurement system shown in Fig. 1 is based on a PC with a standard I/O interface, which provides 16 bit output data forming the address bus, 16 bit bidirectional data realizing the data bus, control lines used for the handshake protocol and three timers. An interface adapter links the I/O port to the system bus. It buffers the data and control bits, supervises the timing for handshaking and renders the system clock.

The system bus is the common interface to all modules. Its timing is defined by a special bus protocol using a handshake procedure. In addition to the I/O bus, it provides the system clock of 4MHz, 8 common clock lines, 8 trigger lines and power supplies. The external clocks are used for controlling the measurement of dynamic events. The modules can be synchronized with respect to rising or falling edges of a clock and are activated by a trigger impulse on a trigger line. This guarantees correct timing and interaction of multiple modules. Depending on the given task the types are chosen and plugged into the slots of the system bus. The modules can be categorized as follows:

- **static modules** writing (reading) digital and analog data to (from) the test object.
- **dynamic modules** writing (reading) digital or analog data to (from) the test object.
- **control modules** controlling the system and the synchronisation of dynamic functions.

The system is able to address 32 different modules and 64 cards of the same type. The test object, which was here the fabricated DTCNN chip [5], is placed on an own routing board and connected to the inputs and outputs of the device.

2.2 Realisation of the Hardware

The different modules are now described in detail:

Static Functions

Digital Out: This module writes 16 binary outputs to the test object. They are transferred from the PC and stored in a register.

Analog Out: This module writes 2 analog output channels within a range between 0V and 5V to the test object. They are transferred binary with 8 bits for each channel from the PC and are stored in a register. Digital data are transformed into analog ones by two 8 bit D/A converters having a resolution of 18 mV.

Digital In: This module reads 16 binary signals from the test object, which are transferred to the PC.

Analog In: This module has 16 analog input channels, which can be read between 0V and 5V. Two of them are selected via 8-channel multiplexers and. The analog signals are applied to 8 bit A/D converters and transferred to the PC.

Dynamic Functions

Digital Out Stack: This module writes 16 binary outputs to the test object, which are loaded by the PC. In contrast to the Digital Out module, the values are preliminary stored in a stack (2048 x 16 bit). Synchronized to a trigger signal, the module starts to issue the data with a rate up to 10 MHz. The trigger signal is selected out of the 8 trigger lines, the output rate is chosen out of the 8 common clock lines. Three different operating modes, namely the single, cyclic, and static mode, do exist. The data block with up to 2048 values can be issued once or cyclic, where the stack is read out periodically. In the static mode it works like the Digital Out module.

Analog Out Stack: This module writes 2 analog output channels within a range between 0V and 5V to the test object, which have been written by the PC into a stack (2048 x 16 bit). It works in the same way like the Digital Out Stack, only the binary data are converted by two fast 8 bit D/A converters.

Digital In Stack: This module starts synchronized with a trigger signal reading 16 binary inputs from the test object with a speed rate up to 10 MHz. The trigger signal can be selected out of the 8 trigger lines, the processing speed can be choosen out of the 8 common clock lines. The input data are stored in a stack (2048 x 16 bit) and are transferred to the PC after the reading operation. The status empty, half full and full can be checked by the PC. If the stack is full or if the first value is read, further input data will be ignored.

Clock Module: This module selects two of the 8 common clock lines and issues a dual phase clock to the test object. The output clock can be switched off and on. It also allows an activation synchronized with the selected trigger signal that begins with the low or the high edge of the chosen clock.

Control Functions

Trigger Module: This module issues the trigger signal (250 ns low impulse) for synchronization. It is provided after a low or a high edge, respectively, on the specified clock line. The reaction can be delayed up to 128 μs in steps of 250 ns.

Reset Module: This module generates a system reset to force a defined state of all modules connected to the bus and to set all outputs to 0V. This is obtained by a power on reset, a software reset or a reset button. It also provides the inputs for the common clock lines.

3 Software Structure

To support the hardware a special test program language TPL has been developed. Its concept is illustrated in Fig. 2.

It can be considered as an interpreter, which processes the statements of the test program step by step. Before the interpreter becomes active, a compiler checks for syntax errors and substitutes the signal names by the real hardware addresses. Every TPL program consists of two parts. The first one defines the connected modules and gives the reference between the logic signal names of the test object and the assigned hardware channels of the testing device in a cross table. The second part contains the executing statements in a Pascal-like notation.

The system enables interaction with the user via the keyboard or available data files. The output is written on the screen. Also graphics are possible enabling a clearly arranged presentation of the obtained results. The data can be stored in files, which allows postprocessing with commercial software packages such as MATLAB or MATHEMATICA.

Below, some examples of statements are given to demonstrate the features of the test set. The statements are grouped after their functionality.

I/O-statements: PUT signals; GET signals; FPUT signals ; FGET signals; CLEAROUT signals;

While the statements PUT/GET write and read signals to the modules Digital In/Out and Analog In/Out, FPUT and FGET enable to write/ read a list of data in a stack for dynamic testing. CLEAROUT resets the stack.

Timing and trigger statements: TRIGGER module, clock, number, edge, delay; STARTOUT signals, clock, trigger, mode; STOPOUT signals;

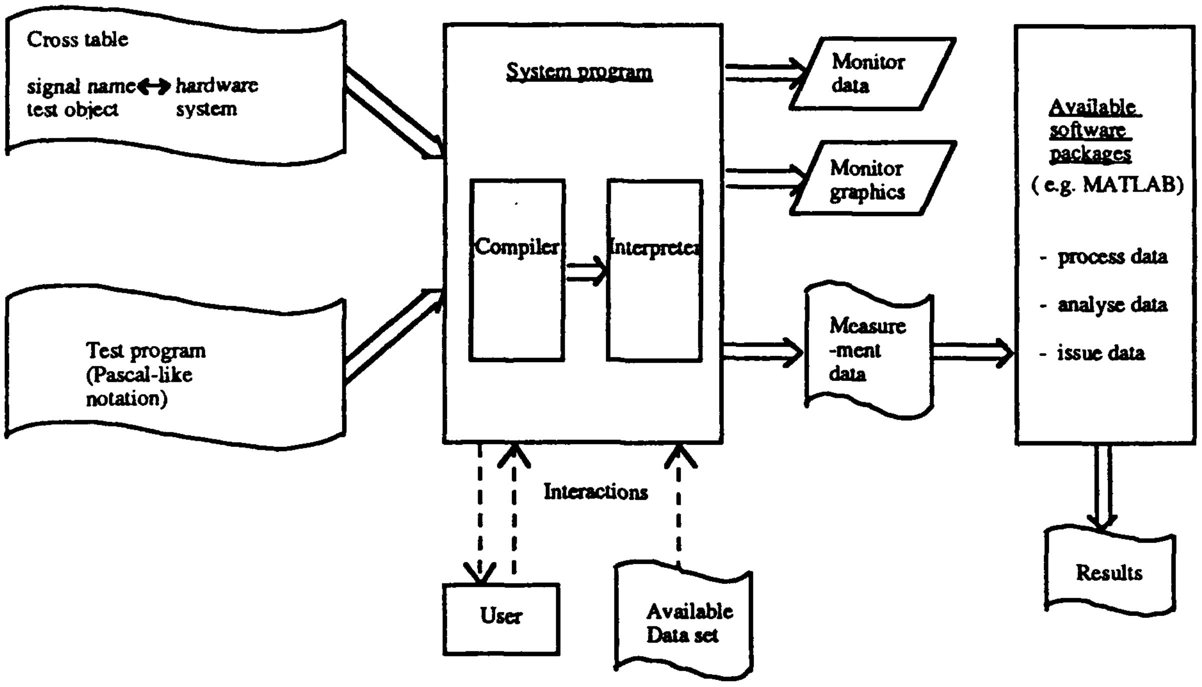


Figure 2: Software concept of the PC-controlled testing device.

TRIGGER generates a trigger signal on the bus line number by a specified module. The signal is synchronized either with the rising or falling edge of a specified clock and can be delayed. STARTOUT activates a stack operation, which is started by a specified trigger pulse, for dynamic signals to a given clock. The mode enables starting, stopping or a synchronized operation with the rising or falling edge of the clock. STOPOUT stops the stack operation of dynamic signals without a reset of the stack.

Programming statements: PROCEDURE BEGIN END, FOR DO DONE, DO UNTIL DONE, IF THEN ELSE ENDIF

The statements correspond to the usual Pascal syntax as do the arithmetic operators.

File statements: RESET file; REWRITE file; APPEND file; READ file, signal; WRITE file, text, signal; CLOSE file;

The TPL enables access to so-called protocol files. Such a file starts with the name PROT and has as an extension a number between 0 and 999. A PROT file is opened for reading by the statement RESET or for writing by REWRITE, respectively. APPEND allows the appending of data. Data are handled by the READ and WRITE statement, where $file = -1$ enables reading from the keyboard or writing on the screen.

Graphic statements: GRON xmin, xmax, ymin, ymax; GROFF; GRPLOT x,y; GRLINETO x,y;

GRON switches the graphic mode on and defines the scaling. GRPLOT draws a point x, y and GRLINETO a line to a new point x,y from the last point defined by GRPLOT or GRLINETO.

The language is expandable for new modules and any C-routines can be embedded. The statements are also available in a C-library, which allows the inclusion in any C-routines.

4 Conclusion

An analog and digital test set has been introduced providing a very comfortable and automatic testing of CNN chips. The hardware is structured into single modules, which communicate via a common system bus to a host computer. Depending on the test object, the system configuration can be chosen just by putting the requested cards into the slots of the system bus. The applied modules must be specified in a definition part, where also the reference between signal names and their assigned hardware channels must be defined. A test program language based on an interpreter has been developed that uses a notation similar to Pascal. Thus a variety of features has been achieved with only a small set of statements, which is easy to use.

References

[1] H. Harrer, *Discrete-Time Cellular Neural Networks*, Dissertation, Technical University Munich, to appear in 1992.

[2] Immanuel Köhn, *Entwicklung eines PC-gesteuerten Testmeßplatzes*, Diplom thesis, Inst. for Network Theory and Circuit Design, Technical University Munich, June, 1992.

[3] NEC Data Book, μPD 8255A, pp 7.69-7.77.

[4] *DCI SmartLab 8255/8253 I/O Card, Operation Manual* (July 1990).

[5] H. Harrer and J. A. Nossek, *New Test Results of a 4 by 4 Discrete-Time Cellular Neural Network Chip*, Proc. of the CNNA 92, Munich, to appear in 1992.

A geometrical approach to the analysis of the dynamical behaviour of neural networks with simple piecewise linear limiters.

Jeroen Dehaene and Joos Vandewalle,
Katholieke Universiteit Leuven,
Department of Electrical Engineering, ESAT-SISTA
Kardinaal Mercierlaan 94,
B-3001 Leuven (Heverlee)
Belgium
tel: 32.16.220931, fax: 32.16.221855
e-mail: dehaenej@esat.kuleuven.ac.be

Abstract

This paper discusses neural networks with the simple piecewise linear limiter (2), which owes much of its popularity to its application in the CNN-model. We present a new approach to understanding the dynamical behaviour of Hopfield type and related models with these limiters. The approach allows for a better intuitive understanding of the model and a good assessment of the possibilities of the model.

1 A geometrical interpretation

Consider a time-continuous Hopfield type neural network with piecewise linear limiters as described by

$$\begin{aligned} \dot{x} &= -x + L(y) \\ y &= f(x) \end{aligned} \qquad (1)$$

where

$x(t) \in R^N$ is the state vector at time t
$y(t) \in R^N$ is the output vector at time t.
N is the number of neurons.
$L : R^N \rightarrow R^N : y \rightarrow Ay + b$ is an affine operation, specified by the weight matrix $A \in R^{N\times N}$ and the vector $b \in R^N$.
$f : R^N \rightarrow R^N$:

$$\begin{aligned} f_i(x) &= -1 \text{ if } x_i < -1 \\ f_i(x) &= x_i \text{ if } -1 < x_i < 1 \\ f_i(x) &= 1 \text{ if } x_i > 1 \end{aligned} \qquad (2)$$

f is the projection on the closed hypercube $C = [-1, 1]^N$. A neuron is said to be in the linear region if $-1 < x_i < 1$ and it is saturated if $x_i > 1$ or $x_i < -1$. The neurons

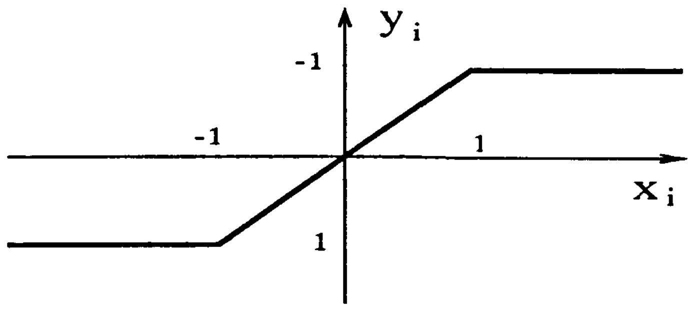


Figure 1: the piecewise linear limiter f_i

on the boundary, $|x_i| = 1$, are assigned to the region they are entering, i.e. to the linear region if $x_i(-x_i + L_i(x)) < 0$, to the saturation region if $x_i(-x_i + L_i(x)) > 0$ and to both if $x_i(-x_i + L_i(x)) = 0$.

The CNN model, as introduced in [1], is a special case of this model with structural restrictions imposed on the matrix A. The input and the independent current sources of the model are both absorbed in the vector b.

Any system

$$\dot{p} = -p + S(p)$$

where $p \in R^N$ is called the state of the system, can be interpreted geometrically in state space as "p follows $S(p)$": The velocity $\dot{p}$ of p is the vector which points to $S(p)$ when based at p. Hence p instaneously evolves towards $S(p)$ with a velocity proportional to the distance. Of course $S(p)$ changes accordingly, such that p evolves like someone following his shadow $S(p)$, for ever or until p and $S(p)$ coincide.

This point of view is especially fruitful for the model (1), with a piecewise linear limiter (2) where the shadow $S(x) = L(y) = L(f(x))$ has a simple geometrical interpretation, leading to intuitive insight in the dynamical behaviour. The output $y = f(x)$ is the projection of the state x onto C. The shadow $S(x) = L(y)$ is the corresponding point (under the affine transformation L) of the parallellepiped $L(C)$.

Fig. 2 shows an easy example for a network with two neurons. A state, initially located in q, starts moving towards its shadow. As the corresponding output, moves along an edge of C, the shadow moves along an edge of $L(C)$. After the output has reached the corner p, the state converges towards a stationary shadow.

2 Benefits of the approach

A classical approach to understand the behaviour of the model with piecewise linear limiters, which has been used in CNN literature [2], considers the system as a piecewise linear system with 3^N different regions, defined by fixing for each neuron whether it is in the linear region, saturated at 1 or saturated at -1. For a region with k neurons in the linear region, the output y lies on a k-dimensional face of C. In each region the system behaves as a (local) linear system, with generically

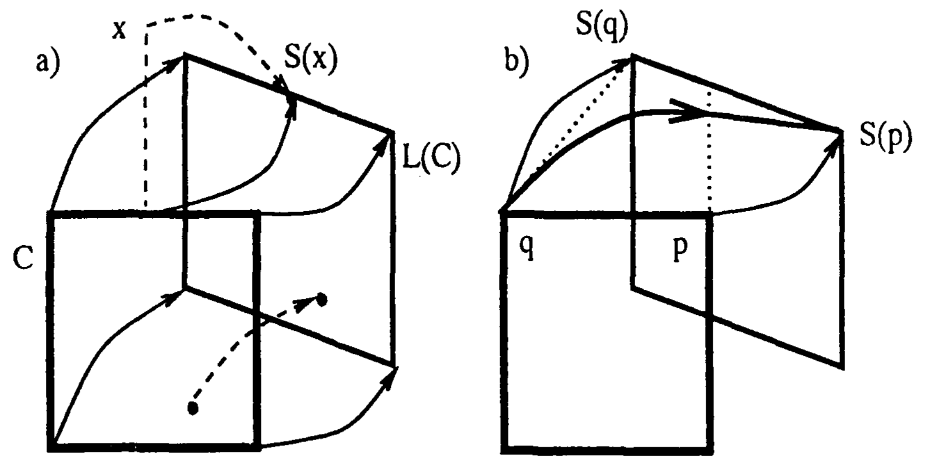


Figure 2: a) the shadow $S(x)$ of a state x, b) a state follows its shadow

as many eigenvalues equal to -1 as saturated neurons (the other eigenvalues can also equal -1 by coincidence). There is a one-to-one correspondence between the 3^N regions and the 3^N subcubes ("hyperfaces") of the hypercube C.

We can easily recover this result by means of the geometrical interpretation, and identify the subspace corresponding to the eigenvalue -1 and the subspace corresponding to the other eigenvalues. The easiest case occurs when all neurons are saturated. All eigenvalues of the local linear system are -1, implying movement in a straight line. In terms of the geometrical interpretation, the *state moves towards a stationary shadow* until either it finds it or it enters another region (the shadow starts moving). If k neurons are in the linear region, the output lies on a k-dimensional face of the hypercube C, and the shadow lies on a k-dimensional face of the parallellepiped $L(C)$. If a state lies in the k-dimensional hyperplane containing this face of $L(C)$, it stays in it. If the state doesn't lie in that hyperplane, it is attracted towards it. The hyperplane is parallel to the eigenspaces corresponding to the $N-k$ eigenvalues (generically) different from -1. The eigenspace corresponding to -1 is orthogonal to the face of the hypercube C.

The shadow approach adds global insight in the dynamical behaviour to the fragmented insight of the piecewise linear approach. To understand the transition in fig. 2b of a neuron from -1 to 1 through the linear region, the piecewise linear approach gives different explanations depending on the eigenvalues of the local linear system (or on the horizontal distance between the shadows). If the system has a positive eigenvalue, the transition of the neuron will be explained as an exponential evolution away from an unstable equilibrium for which the neuron would take a value less than -1. With the negative eigenvalue, there would be attraction towards a stable equilibrium larger than 1. With the shadow approach, one can easily see that the shadow starts at the right hand side of the initial state q, and will stay ahead during the transition to p.

The boundedness of the states, as first proved in [1] follows easily from the fact that the shadow always lies in $L(C)$.

Another merit of the shadow approach lies in the comparison of the model (1) with a related Brain-State-in-a-Box model, which has also been studied in CNN literature [3], :

$$\begin{aligned} \dot{u} &= -u + f(v) \\ v &= L(u) \end{aligned} \tag{3}$$

where $u, v \in R^N$.

It is well known that the associated state $v = L(u)$, behaves exactly according to the equation of the Hopfield model (1). However, although this explains some of the relations between both models, it is still surprising that both models show similar binary input output behaviour, when y is considered for (1) and u for (3). This fact can be explained by observing that u follows a shadow $f(v) = f(L(u))$, which evolves exactly the same way as y in the Hopfield model. Therefore, it is not only possible to map every trajectory of u in the BSB model onto a trajectory of y in the Hopfield model by applying $f \circ L$ but in addition we know that a point of the former trajectory is moving towards the corresponding point of the latter trajectory. It follows immediately that both models have the same equilibria. The attraction basins are generally different, although also this difference is often smaller than expected, since the location of the attraction basins is closely related with the location of unstable equilibria, which are equal for both models.

The main result thus far obtained with the shadow approach is a set of linear inequalities guaranteeing the attraction of all neighbouring patterns (differing in one neuron) to a given pattern. A heuristical solution of these inequalities leads to a new design rule. This will not be treated here since there is no direct connection with CNN. The results can be found in [4] and a more detailed account is given in [5] and [6].

3 A tool for CNN template analysis

The shadow approach can also be a tool in the analysis of templates for CNN. Because of the cellular nature, the behaviour of these models can often be understood by isolating a small part. Consider, for example, the opposite-sign templates as studied in [2] [7] and [8]. The matrix A in (1) is tridiagonal with a constant value p on the diagonal, $-s$ on the upper-diagonal and s on the lower diagonal and $b = 0$. When $(p-1)/2 < s < p-1$, this system behaves as a connected component detector (CCD). If $x_1 = 1$, it converges to an equilibrium with leading neurons 1, followed by alternating neurons 1 and -1, with as many times -1 as connected regions of -1 in the initial image. If $x_1 = -1$, it detects connected regions of 1 in a similar way. Fig. 3a shows the trivial case with two neurons only, where every pattern is stable. For instance, the pattern $[-1 \quad -1]$ should be read as "no components of ones". Now we add a neuron in front of the first two. A crucial observation for the CCD is that the first neuron never changes, once it is saturated. Therefore the effect of adding the first neuron can be described as adding a vector b to the system of the remaining neurons. Therefore, if a -1 is added (indicating ones are counted), the parallellogram of fig 3b, showing the state space for neurons 2 and 3, is located more to the left, such that the pattern [(−1) 1 1] is now unstable and is attracted towards [(−1) −1 1] to be read as "one component of ones". If a 1 is added in front, the parallellogram is shifted down and the system similarly counts components of −1. If an intermediate value, close enough to 0 is added in front, neurons 2 and 3 temporarily behave again as in the case of only two neurons, until the first neuron gets saturated at 1 or −1. One can extend this argument to see that the system behaves as a connected component detector for any number of neurons. Finally, note that the description of the parallellogram shifting up and down, according to the value of a third neuron is nothing else than a two-dimensional representation of the parallellepiped $L(C)$ in the threedimensional state space.

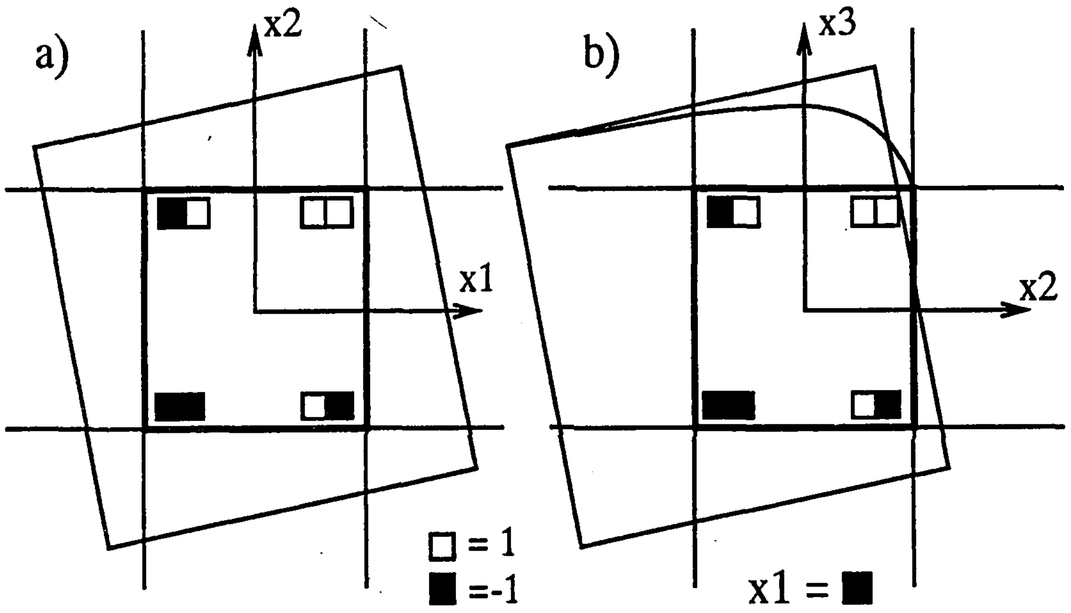


Figure 3: a) CCD with two neurons b) CCD with three neurons

4 Conclusion

A new approach to understanding the dynamical behaviour of Hopfield type and BSB type neural networks was introduced. The method adds intuitive global insight to the often fragmented insight of classical approaches and is useful for CNN template analysis.

References

[1] L.O. Chua, Lin Yang, "Cellular Neural Networks: Theory", IEEE Trans. on Circuits and Systems, vol 35, no. 10, pp. 1257-1272, October 1988.

[2] L.O. Chua, T. Roska, "Stability of a Class of Nonreciprocal Cellular Neural Networks", IEEE Trans. on Circuits and Systems, vol. 37, no. 12, December 1990.

[3] L. Vandenberghe, J. Vandewalle, "Application of relaxation methods to the adaptive training of neural networks.", in 'Proceedings of the 1989 Symposium on the Mathematical Theory of Networks and Systems, MTNS-89, Amsterdam.

[4] J. Dehaene, J. Vandewalle, "A new geometrical approach to the design of neural associative memory", proceedings of the second International Conference on Artificial Neural Networks, ICANN-92, Brighton.

[5] J. Dehaene, J. Vandewalle, "Attraction basins in Hopfield type neural networks, a geometrical interpretation", internal report ESAT - SISTA 1992-8 K.U. Leuven.

[6] J. Dehaene, J. Vandewalle, "A new approach to associative memory for three related neural models", internal report ESAT - SISTA 1992-15 K.U.Leuven.

[7] F. Zou, J.A. Nossek, "Stability of Cellular Neural Networks with Opposite-Sign Templates", IEEE Trans. on Circuits and Systems, vol. 38, no. 6, June 1991.

[8] M. Balsi, "Remarks on the stability and functionality of CNN's with one-dimensional templates", report DNS-5-1991, Hungarian Academy of Sciences.

On The Stability Analysis of Cellular Neural Networks*

F.A.Savacı *   J.Vandewalle **

*Dept.Electrical and Electronics Engineering,İstanbul Technical University,Maslak,80626,İstanbul,Turkey.
**ESAT Lab.,Dept.Electrical Engineering,K.U.Leuven,Kardinaal Mercierlaan 94,B-3001 Heverlee-Belgium.

Abstract. The following results have been obtained on the stability analysis of Cellular Neural Networks(CNNs).
i)By properly defining the Lyapunov function for the CNN we have proved that CNNs with opposite-sign template structure are completely stable for certain values of template values.
ii)By using the 'Toeplitz-Tridiagonal'structure of the state matrices of CNNs with positive-cell linking property we have given some conditions on the template values such that all trajectories of CNNs converge to stable equilibria.
iii)While investigating the conditions for the existence of equilibrium points of CNNs,we have obtained a necessary and sufficient condition for the existence of an equilibrium point in each saturation region,partially saturation region and linear region in which the CNN operates.Actually this condition is valid not only for CNNs,but also for more general types of continuous-time neural network models.Finally,stable and unstable equilibrium points have been determined.

1.Introduction

An analog type neural networks "CNN" with its local interconnection properties have found many application areas and there have been many works on the stability analysis of CNN's [1-6] .In [1], Chua and Yang proved that a network with symmetric weight matrix is completely stable (i.e., every trajectory tends to an equilibrium state). In [2],Chua and Roska attempted to enlarge the stable universe of CNN's by avoiding

*This work was supported in part by the Dept.Electrical Engineering,Katholieke Universiteit Leuven,Belgium.

the restriction symmetry on the weight matrix such as opposite-sign templates with p>1 and conjectured that this type of CNN's are completely stable.But Zou and Nossek ,in [3] proved that for some template values (s>p-1) CNN's with opposite-sign templates have not equilibrium points in the saturation regions which are the only stable regions where stable equilibrium exists. Thus, they showed that complete stability does depend on the template values and the opposite-sign template structure is not itself sufficient for the complete stability.Their emphasize on the existence of equilibrium points in the stable regions is important such that for the symmetric weight matrix case the existence of equilibrium points in the stable regions have yet to be proven.

By defining the Lyapunov function, the conjecture "the CNN with opposite sign-template with s<p-1 is completely stable" given in [3],can be easily proved since s<p-1 already implies the existence of equilibrium in the saturation regions.For the state model of the form

$$\overset{o}{X}=-X + Af(X) \qquad (1)$$

where A has the form as

$$A=\begin{bmatrix} p & -s & & & & \\ s & p & -s & \theta & & \\ & s & p & -s & & \\ & \theta & \ddots & s & p & -s \\ & & & & s & p \end{bmatrix}_{nxn}$$

after defining the energy function as

$$E(t) = -\frac{1}{2} f^{t}(X)\,[A-I]f(X) \qquad (2)$$

where $f(X)=[f(x_1)\ f(x_2)\ldots f(x_n)]^t$, $X=(x_1,\ x_2,\ldots,\ x_n)^t$ and $f(x_i)$ is a piecewise linear function as defined in [1] then it can easily be shown that

$$E(t) = -\frac{(p-1)}{2} \sum_{i=1}^{n} f^2(x_i)$$

which is bounded (-i.e., $|E(t)| < \frac{(p-1)}{2} . n$ since $|f(x_i)| \leq 1$).

The rate of change of energy can be found as

$$\frac{dE(t)}{dt} = -\frac{(p-1)}{2} \Sigma\, 2f(x_i) \frac{df(x_i)}{dx_i} \frac{dx_i}{dt} \quad \text{with} \quad \frac{df(x_i)}{dx_i} = \begin{cases} 1 & |x_i| < 1 \\ 0 & |x_i| \geq 1 \end{cases}$$

which is indeed equal to

$$\frac{dE(t)}{dt} = -(p-1) \sum_{i=1}^{n} x_i \frac{dx_i}{dt} \quad \text{for } |x_i| \leq 1 \qquad (3)$$

From equ.(1) $\dot{x}_i = (p-1)x_i + sx_{i-1} - sx_{i+1}$ and by substituing this into equ.(3)

$$\frac{dE}{dt} = -(p-1)^2 \sum_{i=1}^{n} x_i^2(t) \leqslant 0 \qquad \forall x_i$$

is obtained.

2.SOME REMARKS ON THE STABILITY ANALYSIS OF POSITIVE-CELL LINKING AND OPPOSITE-SIGN TEMPLATES

The remarks given in this section is based on the eigenvalue formula of Toeplitz-tridiagonal matrices given by W.F.Trench [7]. In [2] the state model (1) with

$$A = \begin{bmatrix} p & s & & & \\ r & p & s & \theta & \\ & r & p & \ddots & \\ \theta & & \ddots & \ddots & s \\ & & & r & p \end{bmatrix} \qquad \begin{matrix} r,s > 0 \\ p > 1 \end{matrix} \qquad (4)$$

is considered and by remark 3 it was proved that the solution of such a dynamical system , for almost all initial conditions converges to stable equilibria which are in the saturation regions. This result is the direct consequence of the theorem (4.1) given by Hirsch in [8] .Actually the system (4) with r.s>0 defines a completely stable system according to the theorem given by J. Smillie [9] .

In the linear region,above system corresponds to the

$$\overset{o}{\underset{\sim}{X}} = \begin{bmatrix} p-1 & s & & & \\ r & p-1 & s & \theta & \\ & \ddots & \ddots & \ddots & \\ \theta & & \ddots & \ddots & s \\ & & & r & p-1 \end{bmatrix}_{n\times n} X$$

and its eigenvalues are

$$\lambda_q = (p-1)+2\sqrt{r.s}\cos\left(\frac{q\pi}{n+1}\right),\quad 1\le q\le n \qquad (5)$$

i) p>1 ensures that there exists a positive eigenvalue and hence almost all solutions tend to leave the linear and partial saturation regions towards to the saturation regions.

ii) If we consider case I "i.e., $(p-1)>2\sqrt{r.s}$, r.s>0 then all the eigenvalues are positive and all the solutions of system (4) tends to complete saturation regions.

iii) Considering case II "i.e., $(p-1)<2\sqrt{r.s}$, r.s>0, p>1 and

$$\cos\left(\frac{\pi}{n+1}\right) < \frac{(p-1)}{2rs}$$ " then all the eigenvalues are positive and all the solutions converge to the stable equilibria in the saturation regions if they exist.

iv) Case III implies that almost all solutions will leave the linear and partial saturation regions <u>even if p<1</u> since there exists a positive eigenvalue.

The above argument shows that by arranging the template values we can obtain a completely stable system such that all the solutions of (4) converge to the equilibrium points in the saturation regions and case (iv) also shows that almost all solutions converge to the stable equilibrium points in the complete saturation regions <u>even if p<1</u> as we already know that p>1 is not a necessary condition for the stable equilibrium points being in the saturation regions but only a sufficient condition.

The formula (5) also confirms the results obtained by Roska that the real part of all the eigenvalues of system (4) with r=-s" i.e., opposite-sign template system" in the linear and partial linear regions are strictly positive since

$$\lambda_q=(p-1)+js\cos\left(\frac{q\pi}{n+1}\right)$$

and $p>1$.

3.Conditions For Existence of Equilibrium Points

Let's consider the dynamic equation (1). The external inputs are not considered because the system with constant input can always be transformed into this form by the method given in [10].

<u>Theorem1</u>: There exists an equilibrium point in each of 3^n regions (including linear,partial saturation regions and complete saturation regions) if and only if the matrix A-I is diagonally dominant.

<u>Corollary</u> : If there exists a row (index l) such that

$$a_{ll}-1<-\sum_{j\neq l}^{n}|a_{lj}| \text{ (which indeed implies } a_{ll}<1) \qquad (6)$$

then there doesn't exist any equilibrium point in any saturation region.

<u>note 1</u>: In [11], it has been proved that if inequality is satisfied for every $l=1,2...n$ (i.e., if "I-A" is strictly diagonal dominant with $a_{ii}<1$) then there exists unique stable equilibrium point.

<u>Conjecture :</u> If A-I is strictly dominant then a trajectory corresponding to the system (1) converges to an equilibrium in one of the saturation regions (i.e., the systems (1) is completely stable).

References

[1] L.O. Chua and L.Yang, "Cellular neural networks:Theory, "IEEE Trans. Circuits Syst., Vol. 35, pp.1257-1272, 1988.

[2] L.O. Chua and T.Roska, "Stability of a class of nonreciprocal cellular neural networks, "IEEE Trans. Circuits Syst., Vol.37, pp.1520-1527, 1990.

[3] F.Zou and J.A. Nossek, "Stability of cellular neural networks with opposite-sign templates", IEEE Trans. Circuits Syst., Vol.38, pp.675-678, 1991.

[4] L.O.Chua and C.W. Wu, "On the universe of stable cullular neural networks, "ERL Memorandum UCB/ERL M91/31, Univ. of California, Berkeley, 1991.

[5] C.Güzeliş and L.O. Chua, "Stability analysis of generalized cellular neural networks", to be published in Int. Journal of Circuit Theory and, Applications.

[6] M.Balsi, "Remarks on the stability and functionality of CNN's with one-dimensional tempalets" Report No. DNS-5-1991, Dual and Neural Computing Systems Laboratory, Hungarian Academy of Sciences, Budapest.

[7] W.F.Trench,"On the eigenvalue problem for Toeplitz band matrices",Linear Algebra and it applications.,vol.64,pp.199-214,1985.

[8] M.W.Hirsch,"Systems of differential equations that are competitive or cooperative II:Convergence almost everywhere" SIAM J.Math.Anal.,vdl.16,pp.423-429,1985.

[9] J.Smillie,"Competitive and cooperative tridiagonal systems of differential equations",SIAM J.Math.Anal,pp.530-534,1984.

[10] J.H.Li,A.N.Michel and W.Porod,"Analysis and synthesis of a class of neural networks:Linear systems operating on a closed hypercube,"IEEE Trans.Circuits Syst.,vol.36,pp.1405-1422,1989.

[11]L.Vanderberghe and J.Vandewalle,"Brain-state in a box neural networks with asymmetric coefficients" in Proc.Int.Joint.Con on neural networks pp.627-630,June 1989.

CELLULAR NETWORKS OF OSCILLATORS

Barone A., Balsi M., Cimagalli V.

Dipartimento di Ingegneria Elettronica
Universita' di Roma "La Sapienza"
via Eudossiana, 18
Roma, Italy I-00184

Abstract. The problem of using dynamical equilibria of a Cellular Neural Network (CNN) of relaxation oscillators is faced. The behavior of a planar matrix and of a two-torus of connected oscillators is investigated by computer simulation. Some theorems are stated in the case of: i) two oscillators, ii) a tree of oscillators, iii) a loop of oscillators.

Introduction. It is well known that among neural networks, CNN's [1] exhibit the advantage of possessing only local connections. This property is important with respect to their implementation by means of dedicated integrated circuits.

On the other hand, some recent studies put into evidence that many biological systems behave as distributed oscillatory systems i.e. large arrays (generally planar) of coupled oscillators. A good conjecture, based on results of experiments performed on the visual cortex of cats and the olfactory cortex of rabbits [2,3], seems to be that *synchronization* on different attractors emerging from a chaotic substratum could be an efficient way of storing and processing information.

These two facts induced us to start studying a new architecture possessing these two distinctive features:

1) Its topology is similar to that of a planar Cellular Neural Network with connectivity 1, but with links only along rows and columns of a matrix (no diagonal connections);

2) Neurons are oscillators instead of nonlinear controlled sources. We chose to use relaxation oscillators [4] for two reasons: i) Computer simulations can result more exact as in this case integration is straightforward; ii) Hardware implementation can take advantage of using only digital circuits.

Therefore we study a matrix of connected oscillators, the behavior of which is of course dependent on many parameters: the natural frequency of each oscillator, the strength of each coupling, the initial state of the system. As a natural extension, we will consider also the same matrix wound on a two-torus, even if the analogy with biological structures becomes less evident in this case.

At our best knowledge it is possible to find in literature only studies on the existence of periodic attractors for systems of sinusoidal oscillators with random natural frequencies and equal coupling coefficients [5,6].

In the following, we shall denote as k_{ij} the strength of the coupling between oscillator i and j, i.e., the amplitude of the signal arriving to oscillator i from oscillator j. k_{ii} determines the natural frequency of oscillator i.

Experimental results. Due to the amount of parameters we have to deal with, that is very large even in the case of a small (4 by 4) matrix, as a first step we started using the so called "experimental mathematics". Therefore we performed the following experiments:

A) We considered a planar matrix of 16 identical oscillators (i.e. having the same natural frequency) and found the following types of behavior:

1) In the case when the coupling coefficients are all equal to each other, the behavior of the system is chaotic for every initial state. The Lyapunov exponent, computed by following trajectories in state space and applying Wolf's algorithm [7], seems to be everywhere positive, except when starting from the origin of state space, where Lyapunov exponent is zero.

2) In the case when the coupling coefficients satisfy condition

$$\mathrm{sign}(k_{ij})=-\mathrm{sign}(k_{ji})$$

(opposite-sign condition), the system always reaches a stable oscillatory state at the common natural frequency of the oscillators, with every oscillator differing in phase for 1/4 period from its neighbors (obvious consistency conditions apply). Magnitude of the coefficients is not important.

The stabilizing effect of the oppos: sign structure is analogous to the case of ordinary Cellular Neural Networks [8].

3) With random coupling coefficients, different stable limit cycles are observed, together with non-periodic evolutions: quasi-periodic or chaotic.

In order to inquire into the structure of the attraction basins of the said limit cycles, we performed simulations taking initial conditions along segments in the 16-dimensional state space. Structures of these sets look not at all trivial, and transitions between neighboring basins often seem to be fractal. However, the presence of a set of clearly distinct stable limit cycles, with attraction basins possessing at least a closed inside, makes us believe that they might lend themselves to be used as "codes" or "memory states" for a content-addressable memory (CAM).

The overall structure of attractors is more dependent on the mutual relations between coupling coefficients (e.g. sign, topology...) than on their magnitude. In next section this is proved on simpler structures.

Another set of experiments was performed in order to study the transition between a situation when a stable limit cycle is observed (case 3) and that described under case 1 (transition to chaos). For this purpose, the evolution of the network was simulated for different values of its parameters obtained by moving along a segment in the 48-dimensional space of free coefficients. The results of one of these simulations are shown in figure 1, where the ordinate is the normalized frequency attained at the end of the transient. Zero ordinate means non-periodic behavior.

B) We also considered matrices on a two-torus, which also showed trivial stable behavior under the opposite-sign condition.

In the case of identical coefficients, toroidal matrices show a much larger number of periodic states, with smaller, most probably zero-measure, attraction basins. Two kinds of behavior were observed, the prototypes of which are shown in figures 2 and 3 where the abscissa is taken over a segment, in the 16-dimensional space of initial states, joining two proper sets of them. Figure 2 shows continuous change among indifferent equilibrium limit cycles, while figure 3 shows complex kneading of periodic and non-periodic evolution. As an example of instability, the leftmost state of figures 2 and 3 (the same in the two cases) is periodic, but Lyapunov exponent estimation gives a value of about 1.2.

In the case of coefficients all different from each other, randomly chosen, no periodic evolution has ever been observed.

Theoretical results. We shall consider in this section simpler structures than matrices, in increasing order of complexity. Most of the results described can be straightforwardly extended to sinusoidal oscillator systems.

1) The simplest system consists of two mutually coupled oscillators. Such a system shows a wide spectrum of behaviors: (i) single stable limit cycle, with oscillators operating all at the same or (ii) at different frequencies; (iii) periodic evolution at a frequency dependent on initial conditions: (iv) quasi-periodic or non-periodic motion.

The following results are proved in [9]:

Theorem 1: Necessary and sufficient condition for the two oscillators to evolve at the same frequency is that

$$|k_{12} - k_{21}| \geq |k_{11} - k_{22}|$$

$$k_{22}k_{12} \neq k_{11}k_{21}$$

Duty cycle is 50%.

Theorem 2: If $k_{11}/k_{22} = k_{12}/k_{21} = r$, then the evolution of the system is periodic if r is rational, else quasi-periodic, and the rate of the frequencies of the two oscillators is r.

Theorem 3: If $k_{11} = k_{22}$ and $k_{12} = k_{21}$ (bifurcation condition) the systems evolves periodically with no transitory, because there is an infinity of periodic indifferent equilibrium states.

2) A more complex system is a tree of oscillators. For such a structure it is possible to compute easily a linear relation between the frequencies of the oscillators. The algorithm is described in [9]. It can be extended to a general structure, where also loops are present, but the relation becomes nonlinear, and computational difficulty increases rapidly. The following results are proved in [9]:

Theorem 4: Let

$$D_i^o = 1 - \sum_{j \in \langle i \rangle} R_{ij} D_j^i$$

$$D_j^i = 1 - \sum_{\substack{k \in \langle j \rangle \\ k \neq i}} R_{jk} D_k^j$$

$$R_{ij} = k_{ij} / k_{ji}$$

If all oscillators evolve at the same requency f, and D_o^i 0,then f can be computed through the following recurrent expression, starting from any oscillator i of the tree:

$$f = 0.5\, N_i^o / D_i^o \qquad (1)$$

where

$$N_i^o = k_{ii} - \sum_{j \in \langle i \rangle} R_{ij} N_j^i$$

$$N_j^i = k_{jj} - \sum_{\substack{k \in \langle j \rangle \\ k \neq i}} R_{jk} N_k^j$$

and <i> is the set of indices of all oscillators connected to oscillator i.

If all oscillators have the same natural frequency and $D_o^i = 0$

(bifurcation condition) then (1) takes the form 0/0, and so loses significance; multiple attractors are present (as in the case of two oscillators), depending on initial conditions.

When a loop is considered, the bifurcation condition may be stated as the bifurcation condition on two open chains obtained from it.

It is reasonable to conjecture that the bifurcation condition for matrices corresponds to the fact that any tree that can be extracted from the matrix is in the bifurcation condition.

Conclusion. A planar array of relaxation oscillators has been considered in the view of using it as a new basic structure in the field of Neural Network architectures. Although this goal has not yet been attained, such a structure, never studied before, exhibits a rich and interesting behavior. A systematic research about it was started and some basic results were obtained.

References

1. CHUA, L.O. and YANG, L., *IEEE Trans. on Circ. and Syst.*, 1988, CAS-15(10), pp. 1257-1277
2. GRAY, C.M., KOENIG, P., ENGEL, A.K. and SINGER W., *Nature*, 1989, **338**, pp.334-337
3. SKARDA, C.A. and FREEMAN W., *Behavioral and Brain Sciences*, **10**, pp. 161-195
4. CIMAGALLI, V. and GIONA, M., *Proc. IEEE Int. Symp. on Circ. and Syst. ISCAS '88*, Helsinki, 1988, pp. 262-264
5. SAKAGUCHI, H., SHINOMOTO, S. and KURAMOTO, Y., *Progress in Th. Phys.*, 1987, *77*(5), pp.1005-1010
6. STROGAZ, S.H. and MIROLLO, E., *Physica D*, 1988, **31D**, pp. 143-168
7. WOLF, A., in: HOLDEN, A.V. (Ed.): 'Chaos' (Manchester University Press, 1986)
8. BALSI, M., *Hungarian Academy of Sciences, Budapest*, rep. no. DNS-5-1991
9. BARONE, A., *EE. Dr. dissertation, University of Rome "La Sapienza"*, 1992

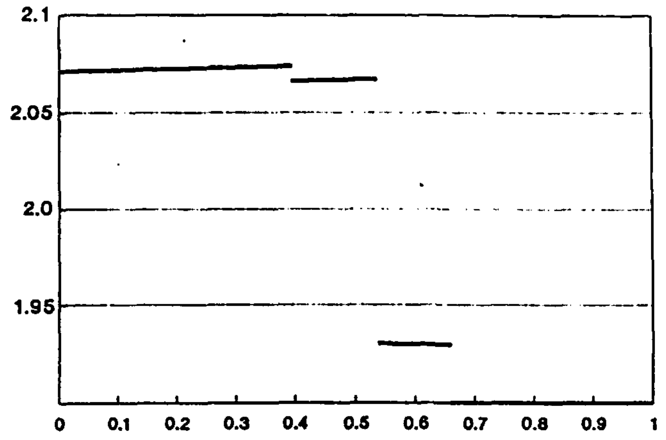


Fig. 1 - Normalized period vs. normalized abscissa on a segment (48-dimensional space) joining two different sets of coefficients, one corresponding to periodic behavior, the other to chaos (bottom ordinate)

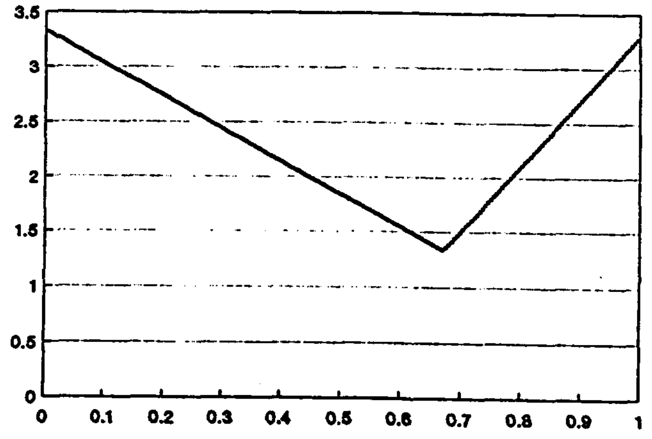


Fig. 2 - Normalized period vs. normalized abscissa on a segment joining two different initial conditions in state space. Infinite periodic attractors of zero measure are shown.

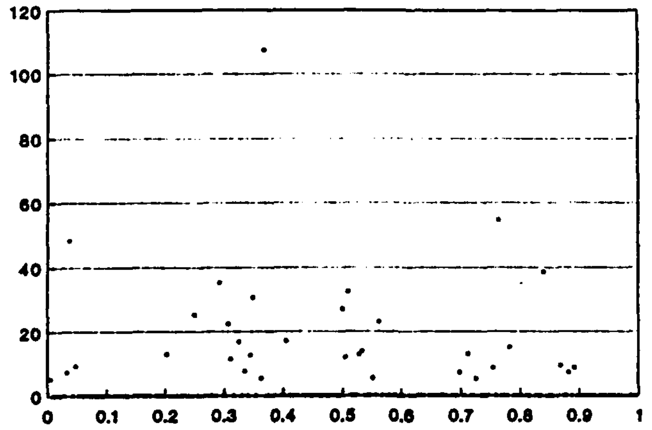


Fig. 3 - Normalized period vs. normalized abscissa on a segment joining two different initial conditions in state space. Mixed behavior of chaos (bottom ordinate) and periodic oscillations is shown.

Discrete-time Cellular Neural Networks using Digital Neuron Model with DPLL

Tomoyuki Ueda, Kiyoshi Takahashi,
Chun-Ying Ho, and Shinsaku Mori

Dept. of Electrical Engineering, Keio University,
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223 Japan
Tel. +81-45-563-1141 Ext. 3319 Fax. +81-45-563-2773
E-mail: tomo @ mori.elec.keio.ac.jp

abstract

In this paper, Discrete-time Cellular Neural Networks using digital neuron model with DPLL (digital phase-locked loop) is proposed. Since consists of all digital elements, it is easy to realize the neuron model by VLSI. We show theoretical analysis and apply it to Shadow Detector in [3] in order to confirm the action of the our neuron model by the computer simulation.

Introduction

Cellular Neural Networks [1][2] represent a new paradigm for nonlinear analog signal processing. In particular, Discrete-time cellular neural networks (DTCNN) have been introduced in [3], which represent an efficient architecture for image processing and pattern recognition. The system can solve global task, although the cell are only locally connected. The simple structure of single cell and the local connectivity make them well suited for VLSI implementation. The realization of DTCNN has been reported in [4] . This has analog circuit structure designed for a CMOS process. However, analog circuits have the following weak points. It is difficult to connect neuro-chips and realize modifiable analog synaptic weight. And the error performance of analog circuits are poorer than that of digital. It is widely expected to realize the DTCNN by all digital circuits.

The neuron model by using multi-input multilevel-quantized digital phase-locked loop (MM-DPLL) has been proposed in [5]. This model has the following advantage. Since it consists of all digital elements, it is easy to realize the neuron model by VLSI. This is the digital neuron model utilizing the phase information. By using the phase, the input signal is modulated by using DPLL, so the model can be applied to the various problems. In [5], it was applied to the Hopfield type associative memory and the pattern recognition. This neuron model satisfies the requirements of the neuron's characteristics such as, spatial summation, temporal summation, thresholding and learning by variable weight. Since the phase modulated signal is widely used in field of the modern digital communications, the phase information type of the digital neuron model will be required in near future. In this study, we propose Discrete-time Cellular Neural Networks using digital neuron model with DPLL and show theoretical analysis. We apply it to Shadow Detector in [3] and confirm the action of the our neuron model by the computer simulation.

Discrete-time Cellular Neural Networks

DTCNN are defined by the algorithm:

$$x^c(t) = \sum_{d \in N_r(c)} a_d^c \, y^d(t) + \sum_{d \in N_r(c)} b_d^c \, u^d + i^c \tag{1}$$

$$y^c(t) = f\left(x^c(t-1)\right) = \begin{cases} 1 & \text{for } x^c > 0 \\ -1 & \text{for } x^c < 0 \end{cases} \tag{2}$$

where the output state $y^c(t)$ of a cell c is binary and is determined by the sign of $x^c(t-1)$. The value of x^c is controlled by the inputs and outputs of adjacent cells d within an r-neighborhood $Nr(c)$. This is defined as the set of adjacent cells within a distance r including cell c. The template coefficients are translational invariant. The value of $i^c \in \mathbb{R}$ is constant and used for threshold.

Digital Neuron Model with DPLL

Before fixing the circuit structure, the grid topology and the neighborhood size have to be determined. We have chosen a 1-neighborhood on the rectangular grid as shown Fig.1, where each cell is connected only to its nine nearest neighbors. In this case, the proposed model as shown in Fig.2, consists of eighteen multilevel-quantized phase comparators(P.C), eighteen random walk filters (RWFi (i = 1~18)), a random walk filter (RWFc), a digital voltage controlled oscillator (Digital V.C.O.), a binary-quantized phase detector (P.D.), and a phase sifter. These are all digital elements. We treat the phase modulated signal as input. The phase of input signal leads (or lags) the reference phase by phase difference θ_{yd} and θ_{ud} [rad]. So the input of this model can be expressed as analog value.

Each multilevel-quantized phase comparator (P.C.) compares the corresponding input θ_{yd} and θ_{ud} with the output phase $\phi(t)$ of digital V.C.O., where $\phi(t)$ is initially set to be equal to the reference phase. As shown in Fig.3, the phase difference is quantized by using the internal high-speed clock operating at R-times the loop natural frequency f_0.

A sequence of pulses produced as lead (or lag) signals is fed to up (or down) inputs of each random walk filter (RWFi) in Fig.4, which is an up-down counter of set length Ni. The content of this counter is increased by up inputs and decreased by down inputs. When the content reaches 2Ni or 0, an erase(-1) or add (+1) signal is obtained as the output respectively, and the counter is simultaneously reset to Ni. In short, RWFi divides the sequence of pulses by Ni and acts as the constant multiplier of weight $a^c{}_d$ and $b^c{}_d = 1/Ni$.

The sequence of pulses weighted by $a^c{}_d$ and $b^c{}_d$ from RWFi is summed spatially as follows,

$$\sum_{d \in N_r(c)} a^c_d\left(y^d(t) - \phi^c(t)\right) + \sum_{d \in N_r(c)} b^c_d\left(u^d(t) - \phi^c(t)\right) \tag{3}$$

and the spatial summation is sent to the second RWFc(set length Nc) which keeps the output of digital V.C.O. stable and provide erase (-1) or add (+1) signals. The second RWFc is inserted because a large number of pulses control the output phase $\phi(t)$ excessively.

Digital V.C.O. consists of three parts : fixed frequency oscillator, Phase controller and divider. As shown In Fig.5, the phase controller eliminates one pulse for an erase signal or adds one pulse for an add signal with the fixed frequency oscillator. The output of phase controller is divided by R to provide the output signal synchronized to the sum of the weighted input phase signals.

Binary-quantized phase detector (P.D.) compares $\phi(t)$ with threshold h of the phase decided by phase sifter. When $\phi(t)$ exceeds h, the output of our neuron model is y = 1, otherwise it is -1,

$$y = SGN(\phi - h) = \begin{cases} 1 & \phi > h \\ -1 & \phi < h \end{cases} \tag{4}$$

Our model can be analyzed by linear model shown in Fig.6 because the phase comparison characteristic is linearly approximated at large R (dividing ratio) , where

$R/2\pi$	: phase comparison coefficient
$a^c{}_d,\ b^c{}_d = 1/Ni$	: coefficient of RWFi
$1/Nc$	: coefficient of RWFc
$2\pi f_0/R$	: coefficient of digital V.C.O.
f_0	: the input frequency

The linear differential equation of this model is expressed as

$$\tau\frac{d\phi(t)}{dt} = -\phi(t) + \left(\frac{\sum\limits_{d\in N_r(c)} a^c_d\, y^d + \sum\limits_{d\in N_r(c)} b^c_d u^d}{\sum\limits_{d\in N_r(c)} a^c_d + \sum\limits_{d\in N_r(c)} b^c_d}\right) \tag{5}$$

$$\frac{1}{\tau} = \frac{f_0}{N_c}\left(\sum_{d\in N_r(c)} a^c_d + \sum_{d\in N_r(c)} b^c_d\right) \tag{6}$$

$\phi(t)$ is obtained as

$$\phi(t) = \left(1 - exp\left(\frac{t}{\tau}\right)\right)\left(\frac{\sum\limits_{d\in N_r(c)} a^c_d\, y^d + \sum\limits_{d\in N_r(c)} b^c_d u^d}{\sum\limits_{d\in N_r(c)} a^c_d + \sum\limits_{d\in N_r(c)} b^c_d}\right) \tag{7}$$

At the stationary state of the step response, Eq.(7) becomes

$$\phi(t) = \left(\frac{\sum\limits_{d\in N_r(c)} a^c_d\, y^d + \sum\limits_{d\in N_r(c)} b^c_d u^d}{\sum\limits_{d\in N_r(c)} a^c_d + \sum\limits_{d\in N_r(c)} b^c_d}\right) \tag{8}$$

In Eq.(6), τ is decided by the input frequency f_0 , set length Nc of second RWFc and the sum of weight. By solving τ , we can obtain the time that a neuron reaches the stationary state and set up a discrete time step.

Simulation

In the simulation to confirm the action of the our neuron model, we use Shadow Detector[4]. The two-dimensional output pattern should consist of the shadow created by an illuminated input pattern. Accepting the restriction of a light source only at the right boundary of the grid, the task can be solved by a wave propagation that starts there. The initial state is black and the pattern is used as input for the cells. The wave propagation in the form of transitions from black to white starts from the right boundary and stops, if a black pixel in the input pattern is reached. The remaining black pixels correspond to the shadow. Set length of RWF*i* and weight set in all neuron are as follows:

$$\begin{matrix} N_1 & N_2 & N_3 \\ N_4 & N_5 & N_6 \\ N_7 & N_8 & N_9 \end{matrix} = \begin{matrix} \infty & \infty & \infty \\ \infty & 32 & 16 \\ \infty & \infty & \infty \end{matrix} \qquad \begin{matrix} N_{10} & N_{11} & N_{12} \\ N_{13} & N_{14} & N_{15} \\ N_{16} & N_{17} & N_{18} \end{matrix} = \begin{matrix} \infty & \infty & \infty \\ \infty & 16 & \infty \\ \infty & \infty & \infty \end{matrix}$$

$$a = \begin{matrix} 0 & 0 & 0 \\ 0 & 1/32 & 1/16 \\ 0 & 0 & 0 \end{matrix} \qquad b = \begin{matrix} 0 & 0 & 0 \\ 0 & 1/16 & 0 \\ 0 & 0 & 0 \end{matrix} \tag{9}$$

The input signal is modulated by θ_{yd} and $\theta_{ud} = +\pi/2$ [rad] at the black region (y^d, u^d = 1), and the signal modulated by θ_{yd} and $\theta_{ud} = -\pi/2$ [rad] at the white region (y^d, u^d = -1). When the input frequency (f_0) is 100kHz and Nc = 32, we considered one cycle to 0.01[ms]. Fig.7 shows an input pattern and the output states in 0,2,4,6,8,10 cycle. From this, we can find that our digital neuron model has the correct action. The result shows that MM-DPLL can be applied to the DTCNN.

Conclusions

Our neuron model consists of all digital elements. DTCNN has the simple structure and local connectivity. Discrete-time Cellular Neural Networks using Digital Neuron Model are suitable for VLSI implementation. It is possible to realize the architecture of Discrete-time cellular Neural Networks, in which the internal information is the phase.

References

[1] L. O. Chua and L. Yang, " Cellular Neural Networks: Theory," *IEEE Trans. on CAS*, vol.35, pp. 1257-1272, 1988.

[2] L. O. Chua and L.Yang, " Cellular Neural Networks: Applications," *IEEE Trans. on CAS*, vol. 35, pp. 1273-1290, 1988.

[3] H. Harrer and J. A. Nossek, " Time-Discrete Cellular Neural Networks: Architecture, Applications and Realization," *IJCNN '91 Singapore*, pp. 718-722, 1991.

[4] H. Harrer, J. A. Nossek and F. Zou, " A Learning Algorithm for Time-Discrete Cellular Neural Networks," *IJCNN '91 Singapore*, pp. 718-722, 1991.

[5] M. Tokunaga, I. Sasase and S. Mori, " Digital Neuron Model Using Digital Phase-Locked Loop," *IEICE Trans. on Information and Systems*, vol. E.74, March 1991.

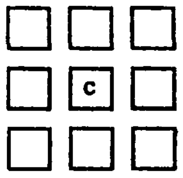


Fig.1: Neighbor cells of cell c for r = 1.

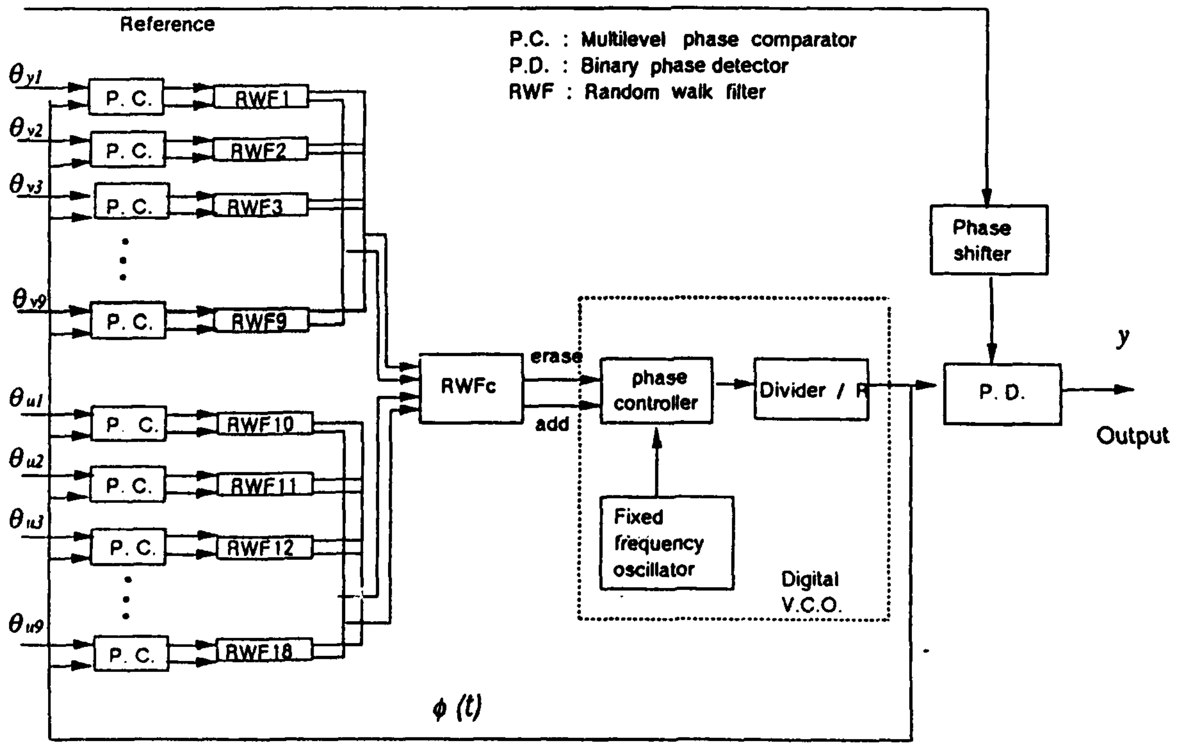


Fig.2: Block diagram of the MM-DPLL

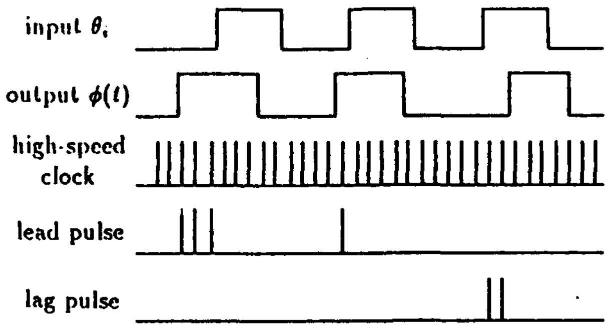


Fig.3: Timing chart of mulilevel-quantized phase comparator

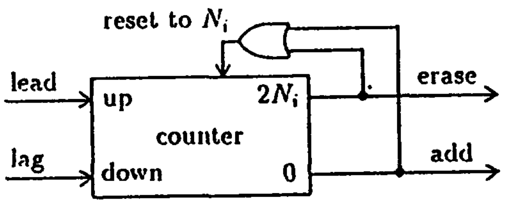


Fig.4: Random walk filter

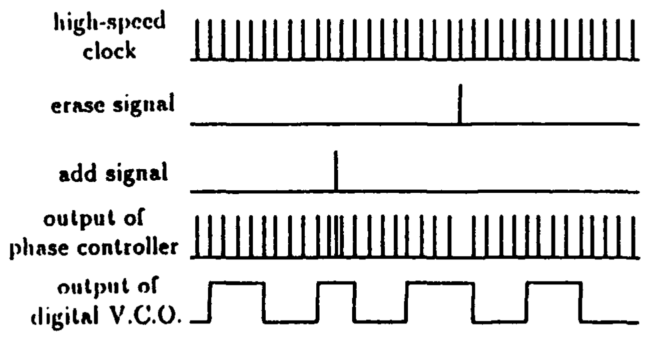


Fig.5: Timing chart of digital V. C. O.

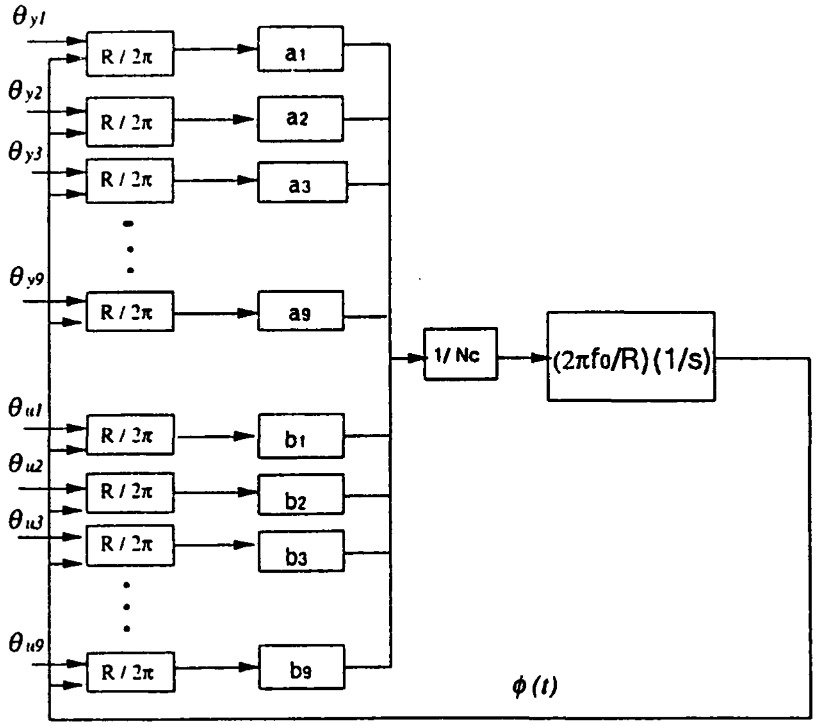


Fig.6: Linear loop model of the MM-DPLL

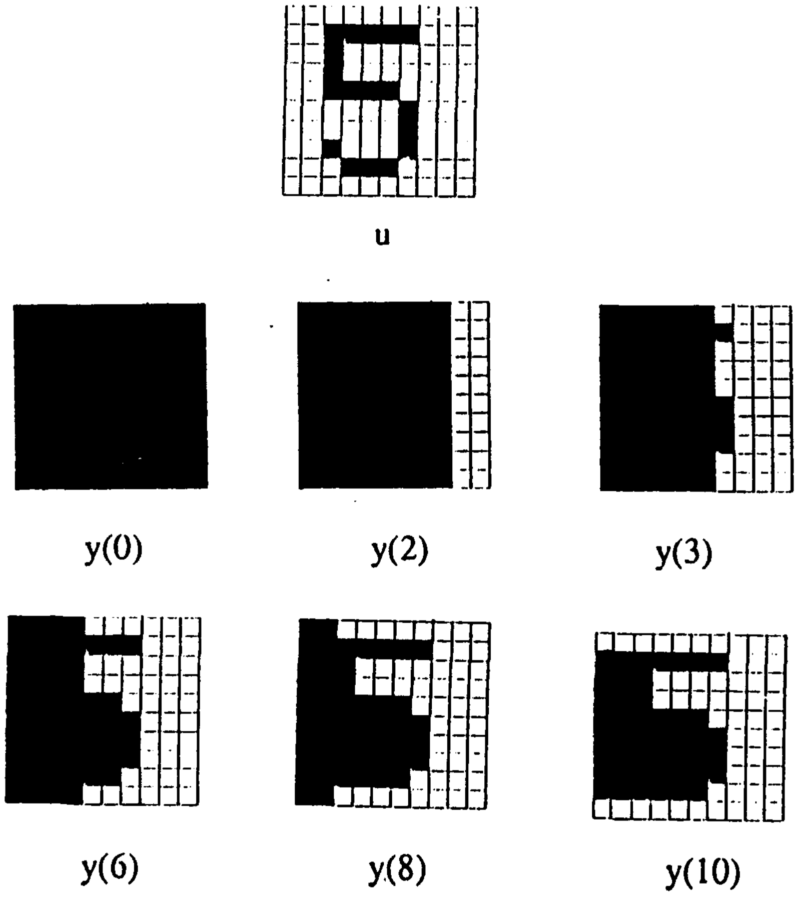


Fig.7: Shadow Detection
Input pattern and output states in our simulation

Symmetry Properties of Cellular Neural Networks on Square and Hexagonal Grids

Gerhard Seiler and Josef A. Nossek

Technische Universität München
Lehrstuhl für Netzwerktheorie und Schaltungstechnik
Arcisstr. 21, D-W-8000 München 2, Germany
Tel.: *49-89-2105 8512; Fax.: *49-89-2105 8504;
E-mail: gese@nws.e-technik.tu-muenchen.de

Abstract

Symmetry properties of cellular neural networks on square and hexagonal grids are investigated: While the symmetry inherent in a given problem should be reflected in the structure of a CNN designed to solve it, the degree of symmetry possible in the network is limited to the symmetry group of the grid it is defined on. The exact numbers of independently choosable entries in both square and hexagonal CNN-templates with different important kinds of symmetry are compared. As expected, the higher symmetry of the hexagonal grid leads to a significant reduction in the complexity of the templates.

1) Introduction

In technology, sampling of spatial information is done on square grids. Nature, on the other hand, organizes its information processing elements in hexagonal arrays. Although the evolutionary success of this arrangement is probably due to the fact that it corresponds to the densest packing of circles in the plane, which arises naturally during cell growth on surfaces, it may also simply be better suited for many problems because of its higher degree of symmetry.

In *Cellular neural networks* or *CNNs* as introduced by Chua and Yang [1], processing elements are also often arranged in a regular grid, and programming for a particular task is done by choosing a set of parameters which are arranged as a regular template. As will be shown, symmetrical tasks pose more stringent constraints on hexagonal templates than on square ones. In practice, this means that hexagonal systems may be easier to build. Symmetry may also be introduced as a design constraint in a more direct design method such as the one proposed in [2].

2) Coordinate Systems and Neighbourhoods

On both grids, coordinate systems must be introduced. To eliminate boundary problems, the grids are supposed to be infinite and plane-filling.

Let $\mathcal{SG}$ be the *square grid* with the cartesian coordinate system, as shown in Fig. 1. Individual cells are denoted by C(i, j), where i is the row index and j is the column index.

The *r-neighbourhood* $N_r^s(i, j)$ of a cell C(i, j) in $\mathcal{SG}$, $r \in \mathbb{N}$ being a positive integer and the superscript s denoting the square grid $\mathcal{SG}$, is defined as in [1] as the following set of cells:

$$N_r^s(i, j) := \{C(k, l) \mid \max(|k - i|, |l - j|) \le r\}. \tag{1}$$

In order to simplify notation, we will also allow neighbourhoods of coordinate pairs.

0-7803-0875-1/92 $3.00 ©1992IEEE

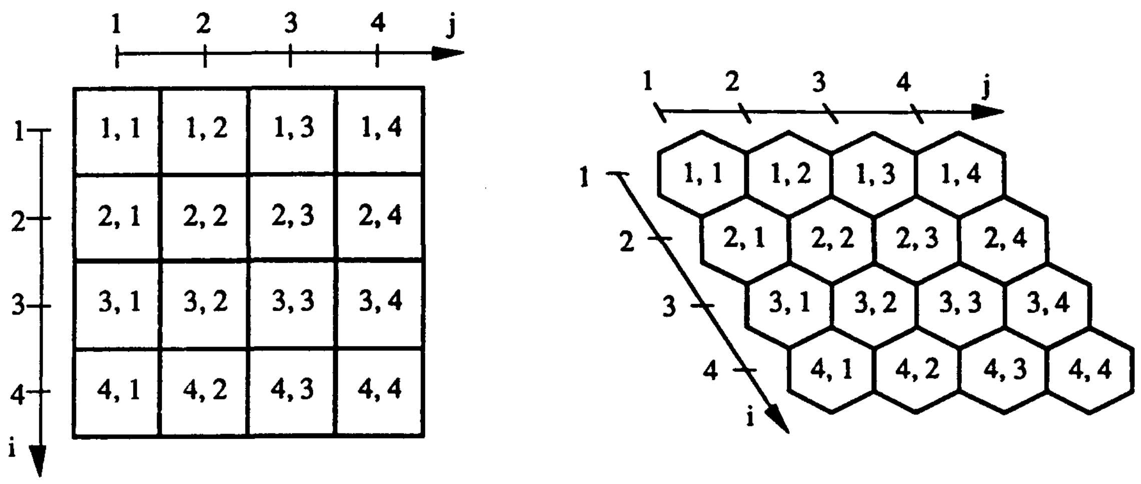


Fig. 1: The square grid (left) and hexagonal grid (right).

Let $\mathcal{HG}$ be the *hexagonal grid*, with the *hexagonal coordinate system* as shown in Fig. 1. Cells are again denoted by C(i, j), where i is the row index and j is the "slanted" column index.

The *(hexagonal) r-neighbourhood* $N^h_r(i, j)$ of C(i, j), where the superscript h indicates the hexagonal grid $\mathcal{HG}$, is the set of (coordinates of) all cells whose centers may be connected to the center of C(i, j) by a curve that intersects at most r hexagon boundaries.

As $\mathcal{HG}$ still has the structure of a lattice, it can be slanted such that the centers of its cells fall on a square grid. The image of $N^h_r(i, j)$ under this transformation is the subset of $N^s_r(i, j)$ that excludes all cells in the top left and bottom right corners whose manhattan distance to the center is larger than r. Therefore $N^h_r(i, j)$ is defined analytically as:

$$N^h_r(i, j) := \{C(k, l) \mid \max\{|k-i|, |l-j|, |(k-i)+(l-j)|\} \le r\}. \tag{2}$$

3) Symmetrical Problems

Symmetry is the invariance of an object under a group of actions. The symmetry groups of most image processing problems are continuous Lie-groups. In electrical engineering systems, however, image data is sampled before being processed on a discrete grid, which leads at best to a discrete subgroup of the original Lie-group. Still, the symmetry inherent in an image processing problem should be reflected as closely as possible by structural symmetry of the processing array.

3.1) Space Invariance

Let M be some space and T(M) the largest group of translations that leaves the structure of M unchanged. Then any image processing problem posed in M that is invariant under T(M) is *space-invariant.*

The translation group $T(\mathbb{R}^2)$ of the real plane $\mathbb{R}^2$ is simply given by

$$T(\mathbb{R}^2) = \{tr: \mathbb{R}^2 \to \mathbb{R}^2 \mid tr(x_1, x_2) = (x_1 + \Delta x_1, x_2 + \Delta x_2) \wedge \Delta x_1, \Delta x_2 \in \mathbb{R}\}. \tag{3}$$

The translation groups $T(\mathcal{SG})$ and $T(\mathcal{HG})$ are isomorphic to $T(\mathbf{Z}^2)$, the restriction of $T(\mathbb{R}^2)$ to the integer lattice $\mathbf{Z}^2$, which performs only integer offsets:

$$T(\mathbf{Z}^2) = \{tr: \mathbf{Z}^2 \to \mathbf{Z}^2 \mid tr(i, j) = (i + \Delta i, j + \Delta j) \wedge \Delta i, \Delta j \in \mathbf{Z}\}. \tag{4}$$

Space invariance of the connection coefficients $a^{k,\,l}_{\ i,\,j}$ implies:

$$a^{k,\,l}_{\ i,\,j} = a^{tr(k,\,l)}_{\ tr(i,\,j)} \quad \forall\, i, j \in \mathbf{Z} \wedge tr \in T(\mathbf{Z}^2),$$

$$\Rightarrow \quad a^{k,\,l}_{\ i,\,j} = a^{k+\Delta i,\, l+\Delta j}_{\ i+\Delta i,\, j+\Delta j} \quad \forall\, \Delta i, \Delta j \in \mathbf{Z}. \tag{5}$$

The connection coefficients in space invariant CNNs therefore depend only on the cells' relative positions. This allows a description in terms of relative positions, the *templates* defined in [1]:

$$a(\Delta i, \Delta j) := a^{i,\,j}_{\ i+\Delta i,\, j+\Delta j} \quad \forall\, i, j, \Delta i, \Delta j \in \mathbf{Z}. \tag{6}$$

If interactions are local, a unique minimal Neighbourhood $N_{r_{min}}(0, 0)$ of the template's origin (0, 0) with minimal finite r_{min} exists, which contains all non-zero coupling coefficients:

$$r_{min} = \min\{r \mid (a(i, j) \equiv 0 \ \ \forall\ (i, j) \notin N_r(0, 0))\}. \tag{7}$$

A template a(i, j) whose elements are only specified for $(i, j) \in N_r(0, 0)$ (and which is assumed to be zero elsewhere) is called an *r-template.*

The numbers $n^s(r)$ and $n^h(r)$ of entries in square and hexagonal r-templates are simply

$$n^s(r) = |N^s_r(0, 0)| = (2r + 1)^2 = 4r^2 + 4r + 1, \tag{8}$$

$$n^h(r) = |N^h_r(0, 0)| = 1 + 6\sum_{i=1}^{r} i = 1 + 3\,r\,(r + 1) = 3r^2 + 3r + 1. \tag{9}$$

For large r, a hexagonal r-template contains 25% less elements than a square one for the same r.

3.2) Isotropy

Let M be some space, and R(M) the largest rotational symmetry group of the neighbourhoods defined in M. Then any space invariant problem defined on M that is also invariant under R(M) is *isotropical.*

The n-element rotation group C_n consists of all n rotations about integer multiples of $\frac{1}{n}\,360°$,

$$C_n = \{E, C_n^{(1)}, C_n^{(2)}, \ldots, C_n^{(n-1)}\}, \tag{10}$$

where E is the unit element and the smallest clockwise rotation $C_n^{(1)}$ generates the group:

$$C_n^{(i+1)} := C_n^{(1)} \circ C_n^{(i)},\ i \in \{1, 2, \ldots, n-1\}, \tag{11}$$

$$C_n^{(n)} \equiv E. \tag{12}$$

Group theory proves, that the invariance of any object under the generators of a group implies its invariance under the whole group. The invariance of a square template under its rotational symmetry group $R(N^s_r(0, 0)) = C_4$ is consequently implied by its invariance under $C_4^{(1)}$, whose action is defined by

$$C_4^{(1)}: a(i, j) \to a(j, -i). \tag{13}$$

Thus, a necessary and sufficient condition for isotropy of square r-templates is

$$a(i, j) = a(j, -i) \quad \forall\ (i, j) \in N_r^s(0, 0). \tag{14}$$

As (0, 0) is the only fixed point of $C_4^{(1)}$, and ali other template entries are mapped on and are therefore identical to exactly three others, the number $n_i^s(r)$ of independently choosable elements on isotropical square r-templates is given by

$$n_i^s(r) = 1 + \frac{n^s(r) - 1}{4} = r^2 + r + 1. \tag{15}$$

Similarly, a hexagonal template is invariant under its rotational symmetry group $R(N_r^h(0, 0)) = C_6$, if and only if it is invariant under $C_6^{(1)}$, whose action in $\mathcal{HG}$ is defined by

$$C_6^{(1)}: a(i, j) \rightarrow a(i + j, -i). \tag{16}$$

A necessary and sufficient condition for isotropy of hexagonal r-templates is therefore

$$a(i, j) = a(i + j, -i) \quad \forall\ (i, j) \in N_r^h(0, 0). \tag{17}$$

As (0, 0) is the only fixed point of $C_6^{(1)}$, and all other template entries correspond to five others, the number $n_i^h(r)$ of independently choosable entries in isotropical hexagonal r-templates is

$$n_i^h(r) = 1 + \frac{n^h(r) - 1}{6} = 1 + \frac{3r^2 + 3r}{6} = \frac{r^2 + r}{2} + 1. \tag{18}$$

For large r, isotropical hexagonal r-templates contain only about half as many independent entries as square ones.

3.3) Complete Symmetry

Let M be some space, and S(M) the (maximal) symmetry group of the neighbourhoods defined in M. Then a space invariant problem on M that is also invariant under S(M) is *completely symmetrical.*

The symmetry group C_{nv} of a regular n-sided polygon consists of C_n and n reflections:

$$C_{nv} = C_n \cup \left\{\sigma_n^{(0)}, \sigma_n^{(1)}, \ldots, \sigma_n^{(n-1)}\right\}. \tag{19}$$

If n is even, the reflections are defined by the orientation of their axes as follows:

- All axes contain (0, 0),
- $\sigma_n^{(0)}$ is horizontal,
- $\sigma_n^{(i+1)}$ is obtained from $\sigma_n^{(i)}$ by tilting it $\frac{1}{n}$ 180° clockwise around (0, 0).

By construction, C_{nv} is shown to be generated by $\sigma_n^{(0)}$ and $\sigma_n^{(1)}$. First,

$$C_n^{(1)} := \sigma_n^{(1)} \circ \sigma_n^{(0)}, \tag{20}$$

which generates the subgroup C_n of C_{nv} as shown in (10) to (12). The other reflections are then obtained as follows:

$$\sigma_n^{(2i)} := C_n^{(i)} \circ \sigma_n^{(0)} \circ C_n^{(n-i)}, \quad i \in \left\{1, \ldots, \frac{n}{2}\right\}, \tag{21}$$

$$\sigma_n^{(2i+1)} := C_n^{(i)} \circ \sigma_n^{(1)} \circ C_n^{(n-i)}, \quad i \in \left\{1, \ldots, \frac{n}{2}\right\}. \tag{22}$$

Hence, a template is completely symmetrical iff it is invariant under $\sigma_n^{(0)}$ and $\sigma_n^{(1)}$ on its grid.

The symmetry axes of square templates are shown in Fig. 2. The actions of $\sigma_4^{(0)}$ and $\sigma_4^{(1)}$ on square templates are defined by:

$$\sigma_4^{(0)}: a(i, j) \to a(-i, j), \tag{23}$$

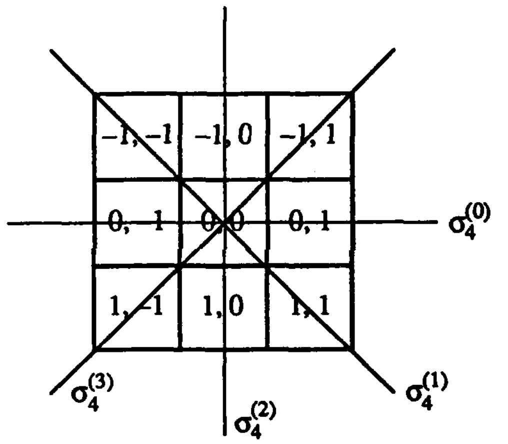
$$\sigma_4^{(1)}: a(i, j) \to a(j, i). \tag{24}$$

A square template is therefore completely symmetrical, if and only if

$$a(i, j) = a(-i, j) \ \wedge \ a(i, j) = a(j, i) \quad \forall\, (i, j) \in N_r^s(0, 0). \tag{25}$$

The number $n_{cs}^s(r)$ of independent entries in a completely symmetrical square r-template is

$$n_{cs}^s(r) = \sum_{i=0}^{r} (i + 1) = \frac{r^2 + 3r}{2} + 1. \tag{26}$$



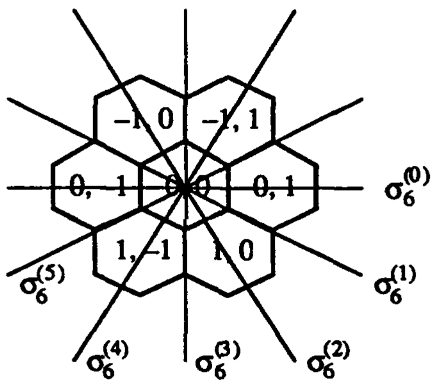


Fig. 2: Possible symmetry axes of square templates (left) and hexagonal (right) templates.

Similarly, Fig. 2 also shows the symmetry axes of hexagonal templates. In hexagonal coordinates, the actions of $\sigma_6^{(0)}$ and $\sigma_6^{(1)}$ are defined by

$$\sigma_6^{(0)}: a(i, j) \to a(-i, i + j), \tag{27}$$

$$\sigma_6^{(1)}: a(i, j) \to a(j, i), \tag{28}$$

and consequently a hexagonal template is completely symmetrical if and only if

$$a(i, j) = a(-i, i + j) \ \wedge \ a(i, j) = a(j, i) \quad \forall\, (i, j) \in N_r^h(0, 0). \tag{29}$$

As the odd numbered axes intersect cells at even distances from the center, but run between pairs of cells at odd distances, one best constructs different formulas for the number $n_{cs}^h(r)$ of independent entries in completely symmetrical hexagonal r-templates for odd and even r:

$$n_{cs}^h(r)\big|_{\text{odd } r} = 2 \sum_{i=1}^{(r+1)/2} i = \frac{r+1}{2}\left(\frac{r+1}{2} + 1\right) = \frac{1}{4}\left(r^2 + 4r + 3\right) = \frac{r^2 - 1}{4} + r + 1, \tag{30}$$

$$n_{cs}^{h}(r)\Big|_{\text{even } r} = n_{cs}^{h}(r+1) - \left(\frac{r}{2}+1\right) = \frac{r^2}{4} + r + 1. \tag{31}$$

With the Gauß bracket []: $\mathbb{R} \rightarrow \mathbb{Z}$, which yields the largest integer smaller than or equal to its argument, one observes that, for odd r,

$$\left[\frac{r^2}{4}\right]_{\text{odd } r} = \frac{r^2-1}{4}, \tag{32}$$

which allows us to merge (30) and (31) into a single final formula for $n_{cs}^{h}(r)$:

$$n_{cs}^{h}(r) = \left[\frac{r^2}{4}\right] + r + 1. \tag{33}$$

For large r, completely symmetrical hexagonal r-templates contain only about half as many independent entries as the corresponding square ones. Also, the total number of independent elements is much smaller than in the isotropical case.

4) Conclusion

For reference purposes, the results on the total numbers of independent entries in unrestricted, isotropical and completely symmetrical square and hexagonal templates are compiled in Table 1.

	square	hexagonal
unrestricted	$4r^2 + 4r + 1$	$3r^2 + 3r + 1$
isotropical	$r^2 + r + 1$	$\frac{r^2 + r}{2} + 1$
completely symmetrical	$\frac{r^2 + 3r}{2} + 1$	$\left[\frac{r^2}{4}\right] + r + 1$

Table 1: The number of independent elements in symmetric r-templates.

The following main conclusions can be drawn:

- Symmetry in a CNN design problem must be considered as early as possible, as its proper exploitation greatly reduce the templates' combinatorial complexity.
- Isotropical image processing tasks are best solved on hexagonal grids, as in this case the savings due to symmetry are much larger than on a square grid.

References

[1] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory", *IEEE Trans. CAS* **35**, 1257–1272 (1988)

[2] G. Seiler, A. J. Schuler and J. A. Nossek, *Design of Robust Cellular Neural Networks*, report TUM–LNS–TR–91–13, Technische Universität München (1990)

[3] J. A. Nossek, G. Seiler, T. Roska and L. O. Chua, "Cellular Neural Networks: Theory and Circuit Design", *Int. Journal of Circuit Theory and Appl.*, to be published in 1992

The Reaction-Diffusion Cellular Neural Network

C.B.Price P.Wambacq A.Oosterlinck
Katholieke Universiteit Leuven
Departement Elektrotechniek
Afdeling ESAT
Kardinaal Mercierlaan 94
B-3001 Heverlee - Belgium

October 1992

Abstract

Reaction-Diffusion equations used to model biological patterning and structure contain useful principles of processing in space; local nonlinear computation plus information spread by diffusion. Some years ago we introduced the use of Reaction-Diffusion Equations as a computational paradigm in image enhancement and analysis, in particular texture-defect analysis. In this paper we desire to bring the Reaction-Diffusion processing paradigm to the attention of the CNN-community. We discuss two classes of processing: First, non-equlibrium systems which form periodic 'Turing' structures allowing analysis of textile images. Second, systems which are able to smooth and segment grey-level images.

1 Introduction

Reaction-Diffusion (RD) systems are nonlinear vector partial differential equations, e.g. the two component system

$$\underline{u}_t = \underline{f}(\underline{u}, \mu) + \underline{\underline{D}}\nabla^2 \underline{u} \tag{1}$$

where $\underline{u}$ is the vector of state variables, $\underline{\underline{D}}$ is the diffusion matrix and μ represents parameters in the nonlinear vector function $\underline{f}(.)$ In a famous paper written to explain the existence of pattern and structure in living systems, Alan Turing showed that such systems could spontaneously generate patterned structures from a stable homogeneous initial state, and that diffusion effects produced the instability [9]. RD-systems are currently proposed to explain patterns from animal coats [3], shells [2] to physical systems [8].

Concerning our image-processing application, let's take the example of textile defect analysis. The basic pattern-forming property of RD systems involves initial growth of a large number of basis pattern modes, e.g., fourier plane-waves, followed by selection of a few interacting modes by the nonlinearity to form a stable pattern. For a given parameter set, several solution patterns may exist. If any of these patterns is similar to an applied noisy textile image with a defect, the system will rapidly move to the closest pattern in all regions of the image except where the defect is. Here the solution will be a different pattern with different numerical values, and leads to easy detection of the defect. This approach was detailed in [4][5].

2 Periodic Structure Formation

Here we shall present the key stages in the analysis of one particular system emphasising details of how it performs as described above. We take the a variant of the Fitzhugh-Nagumo equation (see

[3]), originally used to model nerve pulse propagation, which our analysis shows is the simplest useful nontrivial RD system;

$$\begin{aligned}\frac{\partial u_1}{\partial t} &= f(u_1) - u_2 + D\nabla^2 u_1 \\ \frac{\partial u_2}{\partial t} &= \frac{1}{\epsilon}\left(u_1 - u_2 + \nabla^2 u_2\right)\end{aligned} \tag{2}$$

where the nonlinear function is taken as $f(u) = \alpha u + \beta u^2 - \gamma u^3$, the parameters β and γ allowing us control over the mix of quadratic and cubic nonlinearity. We think of (2) as having the following general form

$$\dot{\underline{u}} = \underline{\underline{J}}\,\underline{u} + \underline{\underline{D}}\,\underline{u} + \underline{N}(\underline{u}) \tag{3}$$

where $\underline{\underline{J}}(0)$ is the linearization of the functions in (2) evaluated at the operating point $\underline{u} = (0,0)$, (to avoid 'excitable behaviour'), and $\underline{N}$ is a nonlinear operator. Consider first the linear problem. If we had been working in continuous physical space with periodic boundary conditions, we should look for a solution as an expansion in plane waves with wavevectors $\underline{k}_j$. The linear operator would become $\underline{\underline{L}} = \underline{\underline{J}}(0) - k_j^2 \underline{\underline{D}}$ and our expansion would be in terms of the vector eigenfunctions of $\underline{\underline{L}}$,

$$\underline{u} = \frac{1}{2}\sum_{n=u,l}\sum_{k} \xi_{nk}(t) e^{i\underline{k}_k\cdot\underline{x}} |n\alpha k> + cc \tag{4}$$

We use the bra-ket notation for clarity, since in general $\underline{\underline{L}}$ is not self-adjoint so we need the left eigenvectors too. Bra's and ket's are parametrized by n, the upper (u) or lower (l) eigenpair of $\underline{\underline{L}}$, α the linear part of the function in (1) which we shall take as a bifurcation parameter, and the spatial frequency k. Equation (4) says the solution is a sum over plane waves each of which is a combination of two vectors giving the contributions of components u_1 and u_2 to $\underline{u}$. Working in the discrete physical space of our CNN, the above expansion holds, only the linear operator must be modified,

$$\underline{\underline{L}} = \begin{pmatrix} \alpha + Dg & -1 \\ \epsilon^{-1} & (g-1)/\epsilon \end{pmatrix} \tag{5}$$

where g is the fourier transform of the discrete diffusion operator. Taking this as coupling to 4 nearest neighbours, (see [6] for details of the 8-neighbour operator), the linear problem $\underline{\underline{L}}|n\alpha k> = \sigma_{n\alpha}(k)|n\alpha k>$ then defines an upper and a lower eigenvalue σ as a function of spatial frequency k. These *dispersion curves* are shown in Fig.1(a). A band of unstable modes is seen which will define the evolving pattern. A full analysis of the linear problem [7] allows us to choose D, ϵ and α to satisfy various requirements additional to this unstable band; no time-periodic solutions, no chaotic solutions and the band peak at a user-definable frequency.

Now let's outline the technique of nonlinear analysis which proceeds by substituting (4) into (3), multiplying by $< n\alpha j|e^{-i\underline{k}_j\cdot\underline{x}}$ and integrating over all space. This yields a series of equations for the time-varying amplitudes, eg, ξ_{uj};

$$\dot{\xi}_{uj} = \sigma_{u\alpha j}\xi_{uj} + \frac{1}{4N_1}\sum_{mn}\sum_{kl}\xi_{mk}\xi_{nl} < u\alpha j|\underline{\underline{N}}_2|m\alpha k>|n\alpha l> + \ldots\text{cubic terms} \tag{6}$$

where $N_1 = < u\alpha j|u\alpha j>$ and

$$< u\alpha j|\underline{\underline{N}}_2|m\alpha k>|n\alpha l> = \sum_{pqr}\frac{\partial^2 f_p}{\partial c_q \partial c_r}(< u\alpha j|)_p(|m\alpha k>)_q(|n\alpha l>)_r. \tag{7}$$

Here the subscripts on the bra's and kets refer to their c_i'th component. Remarkably, for our system (2), the quadratic and cubic nonlinear interaction coefficients evaluate to β and γ respectively. We simplify the amplitude equation set by simple truncation: All lower modes are ignored since their amplitudes are very small due to $\sigma_{l\alpha j}$ being large and negative. Also since our pattern will be produced by the frequency at the centre of the band, we see that harmonic modes can be neglected for the same reason. Fig.1(a) shows that the dc-mode cannot be neglected, so the lowest-order system of amplitude equations describing the formation of a pattern in 2D becomes

$$\begin{aligned}
\dot{\xi}_1 &= \sigma_1\xi_1 + \beta'\xi_2\xi_2^* + \beta'\xi_3\xi_3^* + \beta'\xi_4\xi_4^* - \gamma'\frac{3}{2}\xi_2\xi_4\xi_3^* - \gamma'\frac{3}{2}\xi_3\xi_2^*\xi_4^* - X_1 \\
\dot{\xi}_2 &= \sigma_2\xi_2 + \beta'\xi_2\xi_1 + \beta'\xi_3\xi_4^* - \gamma'\frac{3}{2}\xi_3\xi_1\xi_4^* - X_2 \\
\dot{\xi}_3 &= \sigma_3\xi_3 + \beta'\xi_2\xi_4 + \beta'\xi_3\xi_1 - \gamma'\frac{3}{2}\xi_2\xi_4\xi_1 - X_3 \\
\dot{\xi}_4 &= \sigma_4\xi_4 + \beta'\xi_3\xi_2^* + \beta'\xi_4\xi_1 - \gamma'\frac{3}{2}\xi_3\xi_1\xi_2^* - X_4
\end{aligned} \tag{8}$$

where ξ_1 is the dc-mode and ξ_2,ξ_3,ξ_4 have equal $|\underline{k}|$ shown in Fig.2(b) and where $\beta' = \beta/N_1$ etc. The terms in (8) have been selected by the above analytical procedure since the wavevectors in the corresponding interaction sum to zero, e.g., the term $\xi_2\xi_4$ in the equation for $\dot{\xi}_3$ is selected since $-\underline{k}_3 + \underline{k}_2 + \underline{k}_4 = 0$. Shown in Fig.2 this corresponds to a simple blob pattern. In (8) the X_j represent the sum of self and cross damping terms found in each mode equation, with the form $\xi_j^3/4$ and $\sum_k 3\xi_j|\xi|_k^2/4$, except for ξ_1. These terms always act to bound the growth of all modes, while those terms explicitly shown in (8) cause all modes to grow. Resonances can be seen, eg mediated by the quadratic nonlinearity between modes ξ_2,ξ_3,ξ_4. A detailed discussion of sets of amplitude equations and their properties can be found in [7]. Setting $\xi_2 = \xi_3 = \xi_4$ and assuming that ξ_1 is small, we can solve the equation $\sigma(\alpha) + \beta'\xi - \gamma'\xi^2 = 0$ as a function of α and obtain the solution curve for the blob pattern, Fig.1(b). We have also obtained a solution diagram of a 10-mode truncated system using continuation software. Blobs bifurcate transcritically.

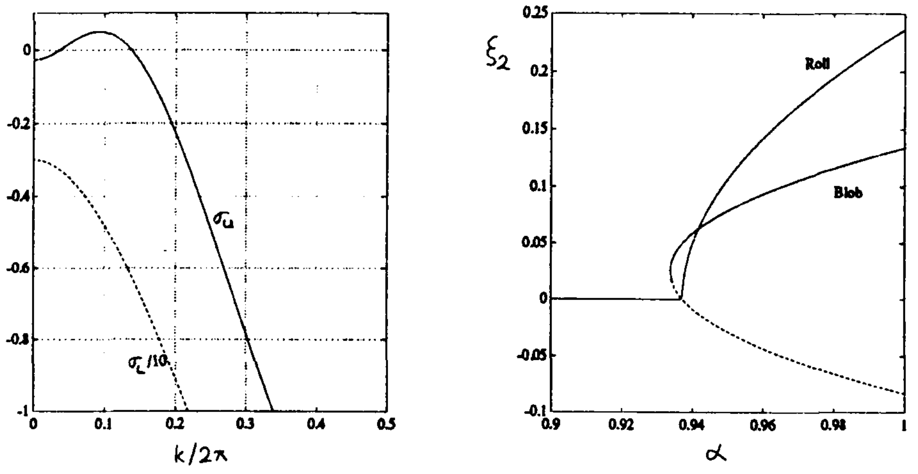
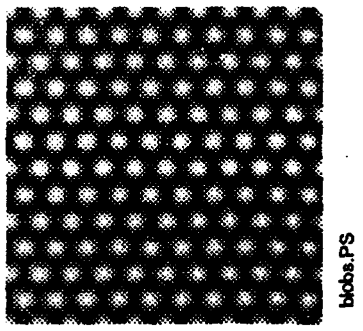


Figure 1: (a) Dispersion curve showing upper and lower σ's, σ_u showing a band of unstable modes.(b) Solution diagram from (8); full lines show stable solution and dotted unstable.

Let's now look how we can use this system to find a defect in a textile image as shown in Fig.3. We assume that we find a range of parameters where the RD-CNN system supports the particular textile pattern and spatial frequency, here we continue with a hexagonal blob pattern. If we apply a textile image with defect, Fig.3(a), to the RD-CNN system, then the modes in (8) and interactions between them will be activated, leading to growth of the blob pattern, at least in areas of correct textile. In the defect region, growth to blob pattern will occur more slowly if at all, since an incorrect mix of spatial frequencies is present (usually too many leading to excessive damping). We have differential growth rates between defect and OK regions so that when the OK



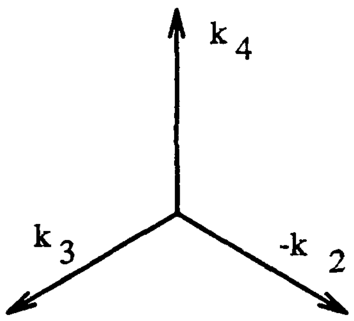


Figure 2: Blob pattern (a) made of three plane waves and (b) their associated wavevectors which sum to zero.

region has reached its 'stable aymptotic state' (SAS), the defect regions, if still evolving, have a low numerical value. Defect detection proceeds by simple thresholding at the OK SAS numerical value determined previously.

The exact details of this differential growth (described in [7]) depend on several factors, one of which is correlation length effects: A wide band of unstable modes implies a short correlation length in space, so that defect and OK areas grow more-or-less independently. If the RD-CNN system supports several patterns for the parameters used, then each region may evolve to an independent pattern each with its own SAS . In our example here, there are two patterns supported, blobs and rolls. Figure 3(c) shows localized evolution to these independent patterns for a short correlation length system. Conversely, when the unstable-mode bandwidth is made narrow, then the correlation length is large, and the differential growth of a single pattern described above occurs. The concept of a parametrically tuned effective correlation length in a CNN is interesting.

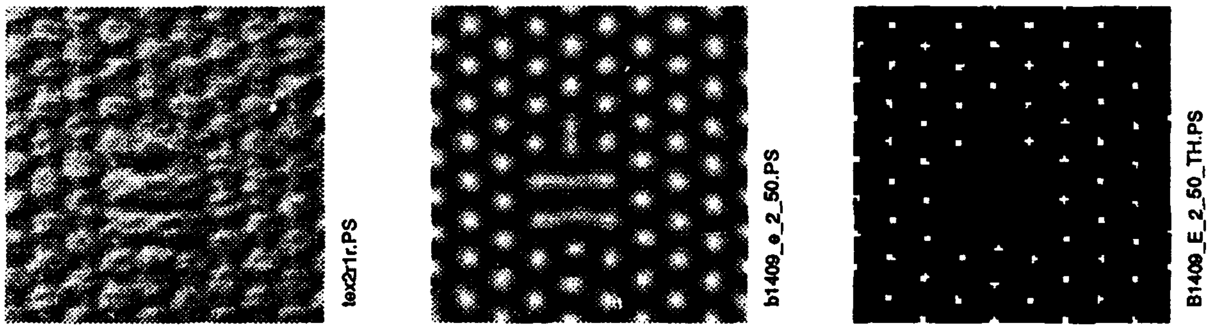


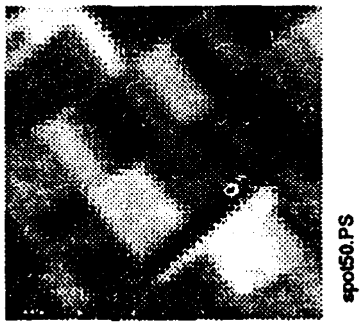
Figure 3: Texture Defect Analysis. (a) Original image (b) SAS (c) Thresholded at pure blob SAS level. The defect is detected as an absence of the correct sized blob. Parameters : $D = 0.5618, \alpha = 0.95, \beta = 0.2828, \gamma = 1.5, \epsilon = 0.25$

3 Image Smoothing and Segmentation

Computer vision requires a segmentation of raw image input into object and background. Classical approaches either maximize regions of similar image information, using principles of continuity and coherence, or else look for edges, using ideas of dissimilarity and compactness. Operators which are able to *smooth noise out of homogeneous regions while preserving discontinuities* are in great demand, as are detectors of lines, curves and other geometrical shapes. Our early investigations of RD systems looked at their ability to produce non-periodic localized solutions, isolated lines and blobs, as would be found in a typical image of blood cells, satellite photos, etc. We found for several systems a useful ability to respond selectively to objects of certain *scale*. There was even evidence of grouping of object parts together in a hierarchical manner. Unfortunately these systems have not yet yielded to analysis, and remain rather *ad hoc*. We nevertheless feel they deserve more attention. Our general approach was to invoke several fields, coupled nonlinearly. Each field, a component in a RD-CNN vector, represented an image property. Our knowledge about the real world and imaging process could be incorporated in the (nonlinear) relations hard-wired into the equations. Let's take a simple example. Consider two fields, $I(\underline{x})$ the image field which is to be derived from the raw data $O(\underline{x})$, and a discontinuity field $g(\underline{x})$. We have suggested the following equation set

$$\begin{aligned} \frac{\partial I}{\partial t} &= -\kappa_I(I-O) + \frac{D_I}{1+\beta g^2}\nabla^2 I \\ \epsilon\frac{\partial g}{\partial t} &= -\kappa_g g + \gamma|\nabla I| + D_g \nabla^2 g \end{aligned} \tag{9}$$

First the discontinuity field $g(\underline{x})$; this is produced by image gradient information $|\nabla I|$, and decays naturally with time constant κ_g and diffuses out with diffusion constant D_g which is usually small. We find that g accumulates close to discontinuities in the image, giving an edge map. Now the image field $I(\underline{x})$; this is produced from the raw input data $O(\underline{x})$ by the first term, and decays naturally with time constant κ_I, which maintains stability. The evolving image data I diffuses to smooth over homogeneous regions, but not over edges, where the large discontinuity field g effectively reduces the diffusion coefficient. Typical results on a satellite image are shown in Fig.4; diffusion is hardly apparent over the strong discontinuties. Other equation sets have been developed including line, junction and corner fields, but at the moment we know of no general mathematical design approach for these systems.



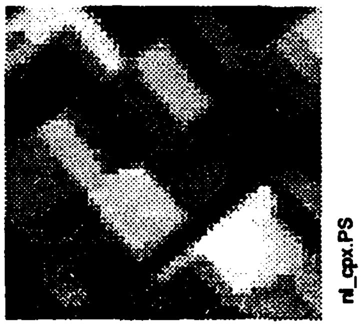


Figure 4: Original satellite agricultural image (a) segmented and smoothed in (b) while preserving discontinuities.

4 Summary

Reaction-Diffusion systems can be designed to have useful cooperative properties enabling them to analyze and enhance grey-value images. In the case of periodic structures, analytical and numerical techniques are advanced enough to allow engineering of systems. We hope similar techniques may soon be found to allow us to replace the *ad hoc* designs we use at the moment with more rigorous systems.

References

[1] E. Doedel, J.P Kernevez,
AUTO - *Software for Continuation and Bifurcation Problems*

in Ordinary Differential Equations
Caltech, 1986

[2] H.Meinhardt, M.Klinger
A Model for Pattern Formation on the Shells of Molluscs
J.Theor.Biol. (1987) 126, 63-89

[3] J.D.Murray
Mathematical Biology
Springer-Verlag (1990).

[4] C.B.Price, P.Wambacq, A.Oosterlinck
Image enhancement and analysis with reaction-diffusion paradigm.
IEE Poc. Col.137, Pt.1, No.3, June 1990

[5] C.B.Price, P.Wambacq, A.Oosterlinck
Applications of Reaction–Diffusion Equations to Image Processing
IEE 3rd Int. Conf. Image Processing and its Applications (1989) 49-53

[6] C.B.Price P.Wambacq, A.Oosterlinck
The Plastic Coupled Map Lattice - A Novel Image-Processing Paradigm
Chaos Vol.2, No.3 1992

[7] C.B.Price
Ph.D Thesis. (Ready soon).

[8] H.G.Purwins, Ch.Radehaus,
Pattern Formation on Analogue Parallel Netowrks
in Neural and Synergetic Computers,
Springer-Verlag (1988).

[9] A.M.Turing
The Chemical basis of Morphogenesis
Philos.Trans.Roy.Soc.London Ser B 237 (1952) 37-72

Mapping Nonlinear Lattice Equations onto Cellular Neural Networks

S. Paul, J. A. Nossek,
Institute for Network Theory and Circuit Design,
Technical University Munich,
Arcisstr. 21, 8000 Munich 2, Germany
and L. O. Chua
Dept. of Electrical Engineering and Computer Sciences,
University of California, Berkeley, CA 94720. U.S.A.

Abstract

In the last years completely integrable Hamiltonian systems were of great interest because of their physical nature, e.g. the existence of soliton solutions, and their relation to eigenvalue and sorting problems. But until recently, they fourd little interest among electrical engineers.

Under certain restrictions, cellular neural networks (CNN) come very close to some Hamiltonian systems, therefore they are potentially useful for simulating or realizing such systems. In this paper, we will show how to map two one-dimensional nonlinear lattices, the Fermi-Pasta-Ulam lattice and the Toda lattice, onto a CNN. We demonstrate for the Toda lattice, what happens, if the signals are driven beyond the linear region of the PWL output function. Though the system is no longer Hamiltonian, numerical experiments reveal the existence of solitons for special initial conditions. This interesting phenomenon is due to a special symmetry in the CNN system of ODE's.

1 Introduction

In recent years, completely integrable Hamiltonian systems have attracted great interest. Closeiy related to them is the existence of soliton solutions. The Fermi-Pasta-Ulam lattice and the Toda lattice are classic examples of such systems [1] [2]. The Hamiltonian structure of such systems implies, among other things, the physical property of losslessness with respect to energy. Due to the discovery of systems, like the Toda lattice, which are capable of solving eigenvalue problems [3], a lot of important applications have become possible. Consequently, any task which can be formulated as an eigenvalue problem is a potential application, e.g. rank filtering [4], [5], [6].

A lot of current interest in electrical engineering has been focussed on regular architectures of analog nonlinear processing cells with local connections, called cellular neural networks [7], [8]. In general, they are not lossless and so soliton solutions do not seem to exist. Recently, Roska et. al. generalized the original CNN structure to allow fairly general nonlinear and delay-type feedback and control [9].

In this paper we demonstrate two things. First we show under what conditions the Fermi-Pasta-Ulam (FPU) lattice can be realized exactly by a conventional CNN. Afterwards we map a transformed version of the Toda lattice [10], [11], [3] onto the current framework of CNN.

The original CNN equations contain a piecewise-linear (PWL) map of the states with a saturation characteristic. This nonlinearity can destroy the integrability of the Toda lattice equations. But surprisingly under certain conditions it is still possible to find soliton solutions. The "invariants" of motion are no longer invariant, i.e. the system is not lossless but they exhibit an interesting recurrent phenomenon. These recurrent phenomenon still allows sorting, subject to some additional interpretations.

2 Cellular Neural Networks

The CNN is a nonlinear dynamical system of autonomous ordinary differential equations (ODE) [7], [8]. If the state variables $v_{xij}(t)$ are arranged in an $M \times N$ rectangle, then only the states in a local neighbourhood of any particular state v_{xij} are coupled to the state v_{xij}. Using the current definition of CNN nonlinear templates $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ which are functions of internal states

and input signals, as well as time delays are allowed. In this paper, however, only delay free polynomial type nonlinearities are admitted. The equations describing the most recent class of CNNs without time delay with $1 \leq i \leq M; 1 \leq j \leq N$ are as follows:
State Equations:

$$\begin{aligned} C\frac{dv_{xij}(t)}{dt} &= -\frac{1}{R_x}v_{xij}(t) + \sum_{C(k,l)\in N_r i,j} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in N_r i,j} B(i,j;k,l)v_{ukl}(t) + \\ & \sum_{C(k,l)\in N_r i,j} \hat{A}(i,j;k,l)\left(v_{yij}(t), v_{ykl}(t)\right) + \sum_{C(k,l)\in N_r i,j} \hat{B}(i,j;k,l)\left(v_{yij}(t), v_{ukl}(t)\right) + I, \end{aligned} \quad (1)$$

Output Equations:

$$v_{yij}(t) = \frac{1}{2}\left(|v_{xij}(t)+1| - |v_{xij}(t)-1|\right), \quad (2)$$

Input Equations:

$$v_{uij} = E_{ij}, \quad (3)$$

Constraint Equations:

$$|v_{xij}(0)| \leq 1, \qquad |v_{uij}| \leq 1, \quad (4)$$

Parameter Assumptions:

$$C > 0, \quad R_x > 0. \quad (5)$$

For the purpose of this paper we will relax some of the assumptions to tailor to our needs. They become clear in the context of the lattice mapping.

3 Lattice Equations

In general, one-dimensional lattices are described by a set n of structurally identical ODEs (first or second order) with local coupling between the states $\dot{x}_k = f(x_{k-1}, x_k, x_{k+1})$. A special case of these lattices are one-dimensional mechanical system consisting of n mass points connected by nonlinear springs, i.e. a system of coupled oscillators. It is governed by the equations of motion:

$$\ddot{x}_k = F(x_k - x_{k-1}) - F(x_{k+1} - x_k) \qquad k = 1, 2, \ldots, n. \quad (6)$$

with boundary conditions at both ends and initial conditions for all mass points.

Quadratic, cubic or piece-wise linear spring characteristics were investigated in the fourties and early fifties by Fermi, Pasta and Ulam [1]. They observed that a sinusoidal initial condition for x_k is repeated periodically, though the system is nonlinear. Related to this recurrence phenomenon is the existence of soliton solutions of the lattice equations.

Toda investigated the discrete one-dimensional lattice with the nonlinear spring characterized by an exponential force vs. r relationship, where r denotes the distance between the mass points [2], $F(r) = \exp(-r) - 1$, and where the outer mass points are fixed at $x_0 \to -\infty$ and $x_{n+1} \to \infty$. It can be shown, both analytically and experimentally, that solitons exist [13]. The equations of Toda were the origin of a series of mathematical developments after Symes' discovery of an intimate relationship between the Toda lattice and the eigenvalue problems of symmetric tridiagonal matrices [3]. With a nonlinear coordinate transform [10] the equations of motion (6) with exponential force can be written as

$$\begin{aligned} \dot{a}_k &= 2\left(b_k^2 - b_{k-1}^2\right) & (7) \\ \dot{b}_k &= b_k\left(a_{k+1} - a_k\right). & (8) \end{aligned}$$

In the following we will call these equations the Toda lattice equations, since they are homeomorphic to the Toda lattice . If the initial conditions for a_k, b_k are specified as the entries of a symmetric tridiagonal matrix $\mathbf{A}(0)$ with diagonal entries a_i and off-diagonal entries b_i, then the fixed point of (7), (8) is characterized by $a_k = \lambda_k$, $b_k = 0$ where λ_k are the eigenvalues of $\mathbf{A}(0)$ with the additional property that $\lambda_1 > \lambda_2 > \ldots > \lambda_n$, i.e. they come out sorted. As is well

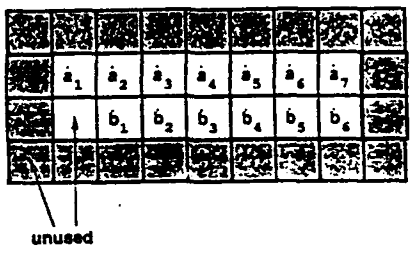

a_1 a_2 a_3 a_4 a_5 a_6 a_7
b_1 b_2 b_3 b_4 b_5 b_6
unused

Figure 1: Mapping of the Toda lattice equations onto a la CNN architecture. The variables in the cells denote the equation that is realized by the respective cell

known, a property of completely integrable Hamiltonian systems is the existence of as many invariants of motion H_k as the degrees of freedom. The invariant H_1 denotes the trace of the matrix A(0) and H_2 denotes the Frobenius norm of A(0). The system posesses $n!$ fixed points according to the $n!$ permuations of λ_k.

4 CNNs and Lattice Equations

After presenting the equations for CNN with unspecified matrices A, B, $\hat{\text{A}}$, $\hat{\text{B}}$ and the equations for the FPU and the Toda lattice, we are now ready to relate both systems. In the case of a first order ODE lattice equation, the states of the lattice equation are identical to the CNN states. The lattice equations from mass spring models are second-order ODEs in time with one spatial dimension, while the CNN equations are first-order ODEs with two spatial dimensions. Hence, we have the choice to use either a double-layer CNN or a $2 \times n$ cell single-layer CNN. A special choice for the template matrices A and $\hat{\text{A}}$ make the structure absolutely regular (Fig. 1 for Toda lattice). This works well under the condition that two rows and columns (shaded area) of cells deliver zero signals to the cells.

4.1 Fermi-Pasta-Ulam Lattice

The FPU lattice with the PWL spring characteristic fits perfectly into the original CNN structure with the coordinate transformation $v_{xij} = x_i - x_{i-1}$ and the template

$$\mathbf{A} = \begin{bmatrix} 0 & K & 0 \\ 0 & 0 & 0 \\ \frac{1}{K} & -\frac{2}{K} & \frac{1}{K} \end{bmatrix}, \tag{9}$$

where all other templates are zero. The scaling factor K has to be adjusted such that v_{x1j} does not leave the linear region of the output function. Only the states v_{x2j} are forced to the linear region of the output map. In this way simple experiments of the FPU lattice can be performed on this CNN. The mapping of the FPU lattice onto the CNN is exact.

4.2 Toda Lattice

The Toda lattice equations can be further simplified by a transformation $c_k = b_k^2$, proposed in [6], and a linear time scaling to drop the factor 2.

In detail the following template matrices are to be implemented:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1/\hat{R}_x & 0 \\ 0 & 1 & -1 \end{bmatrix}, \quad \hat{\mathbf{A}} = \begin{bmatrix} 0 & f_{12} & f_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{10}$$
$$f_{12} = -v_{yij} v_{ykl}, \qquad f_{13} = v_{yij} v_{ykl}.$$

We call this system the modified Toda lattice. The term $1/\hat{R}_x$ is useful only in the presence of $R_x < \infty$. It makes the system less dissipative, but the choice of $\hat{R}_x$ is very restrictive ($\hat{R}_x > R_x$)

for stability reasons. As opposed to the FPU lattice, the mapping of the Toda lattice is exact if all states $v_{x_{ij}}$ utilize only the linear part of the output function. Bounds for the initial values can be given.

4.3 Modified Toda Lattice

Because of the nonlinear output map, the CNN does not exactly behave like a Toda lattice for initial conditions, which not meet the bounds. The saturation characteristics bounds the derivatives, namely $|\dot{a}_k| \leq 2$, $|\dot{b}_k| \leq 2$. This implies bounded acceleration and propagation speed in terms of the Toda lattice. Due to the boundary conditions $b_0 = b_n = 0$, the fixed point is still given by $b_k = 0$. This follows by induction from the boundary conditions and the equations for $\dot{a}_k$. As for the Toda lattice, no information on the a_k is available.

5 Numerical Experiments with Modified Toda Equations

This section summarizes numerical experiments on the modified Toda lattice ((1) - (5) and (10)) with a PWL output map (2). The FPU lattice is not considered here because we only wanted to show that a CNN could be used for an electrical circuit realization of this lattice.

5.1 The Most Simple System

At first 3 examples for a 2×2 matrix are investigated.
Example:

$$A(0) = \begin{bmatrix} -2.0 & 0.1 \\ 0.1 & 0 \end{bmatrix}$$

The transients for a_1, a_2, b_1 are shown in Fig. 2 a, b. The bounds of the derivatives stretch the

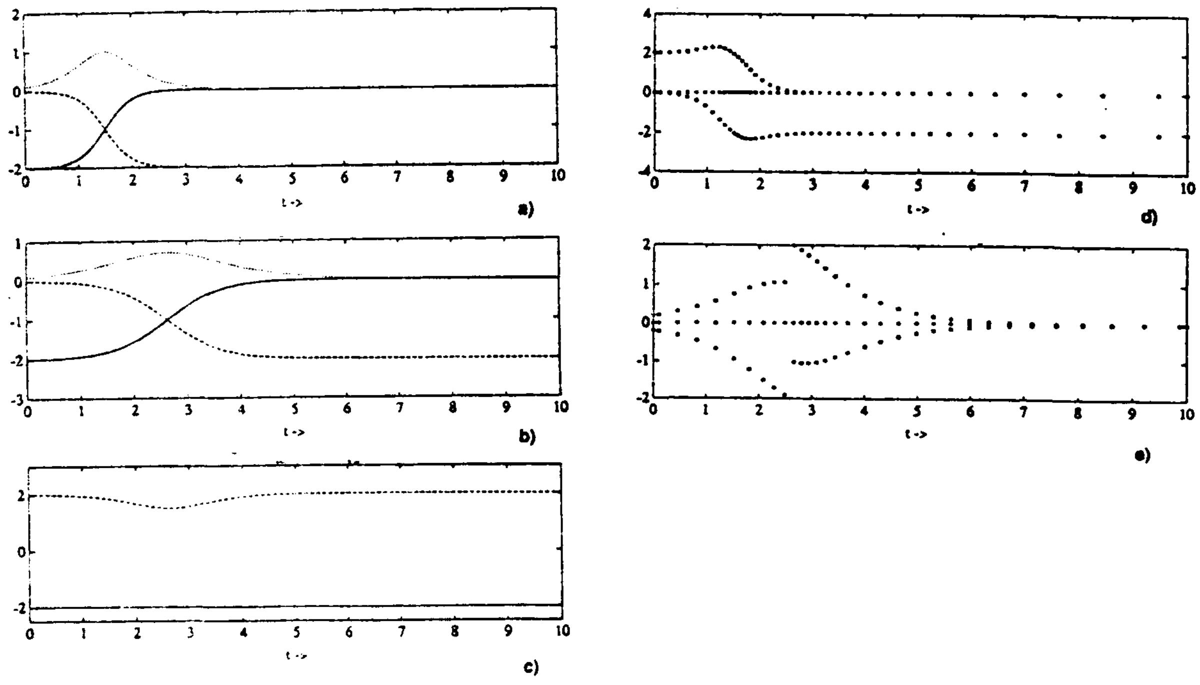


Figure 2: Example, a), b) trajectories a_1 (—), a_2 (- - -), b_1 (......), c) invariants H_1 (—), H_2 (.....), d), e) eigenvalues of the Jacobian matrix

transients, but the initial values $a_1(0)$, $a_2(0)$ are still sorted by value. The system is no longer lossless but can be lossless, passive or active. The passive and active mode keep a balance such that the invariant H_2 is recovered after sorting. The invariant H_1 does not change its value.

The solution of the Toda lattice and the modified Toda lattice differ by a time scaling of $\sqrt{2}$. The invariant H_2 yields $H_2 = \tilde{\beta}^2 + \tilde{\alpha}^2 - \tilde{\alpha}^2/2\cosh^2(\tilde{\alpha}t + \tilde{\gamma})$. Hence, H_2 is not constant, but decreasing for $t_m < -\frac{\tilde{\gamma}}{\sqrt{2}\tilde{\alpha}}$, i.e. the system is passive, and increasing for $t_m > -\frac{\tilde{\gamma}}{\sqrt{2}\tilde{\alpha}}$, i.e. the system is active. So we have a symmetry with respect to t_m.

In the modified Toda lattice for dimension $n = 2$ with special initial conditions such that the ODE is still nonlinear, a recurrence phenomena can be observed. This is due to a symmetry in time, e.g. of the eigenvalues of the Jacobian matrix.

5.2 Sorting of Numbers

The modified Toda equations behave like a sorter, but a strict sorting order no longer exists. This is due to the limited propagation speed of the data and the stability properties of the fixed points. The linearization of the modified Toda lattice about a fixed point gives: $\dot{a}_k = 0$, $\dot{b}_k = (g(\hat{a}_{k+1}) - g(\hat{a}_k))g'(0)b_k$. The variables $\hat{a}_k$ denote the values of the diagonal entries at the fixed point and $g'(0)$ denotes the derivative of the output function. A fixed point is called asymptotically stable[1] with respect to perturbations in b_k, if for all first sub- and superdiagonal entries, the following inequality holds ($g'(0) > 0$): $g(\hat{a}_{k+1}) - g(\hat{a}_k) < 0$.

The inequality gives no information on the order for $\hat{a}_{k+1}$ and $\hat{a}_k$ for $\hat{a}_{k+1} > 1$ and $\hat{a}_k > 1$ (-1 equivalently). So their order at the fixed point is undecided.

Suppose $|\hat{a}_k| < 1$ for n_s values, $\hat{a}_k > 1$ for n_+ values and $\hat{a}_k < -1$ for n_- values. Then, for the modified Toda lattice the number of asymptotically stable fixed points n_{fp} is determined by the number of permutations of the elements with $\hat{a}_k > 1$ and with $\hat{a}_k < -1$, thus $n_{fp} = n_+!n_-!$.

If the output map is applied to the sorting result, a half order is realized, since all values $\hat{a}_k > 1$ ($\hat{a}_k < 1$) reduce to $\hat{a}_k = 1$ ($\hat{a}_k = -1$).

The multiplicity in the fixed points is caused by the bounded derivatives. Therefore the propagation speed for data $|\hat{a}_k| > 1$ is equal. So their final position depends on the initial position.

5.3 Soliton Solutions

An interesting result of the modified Toda lattice equations is the existence of soliton solutions. Such a solution requires the initial conditions to be such that the ODEs operate in a nonlinear regime. Then a wave with constant shape is built up and two waves pass each other without changing their shape after the collision.

In Fig. 3 the evolution of (states vs. time) of the variables a_k for a lattice of dimension 7 with initial values

$$\begin{bmatrix} a_1, & \ldots, & n-1, & n \\ b_1, & \ldots, & b_{n-1} & \end{bmatrix} = \begin{bmatrix} -0.75 & 0 & 0 & 0 & 0 & 0 & 1.4 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \end{bmatrix}$$

is shown. The little irregularity in the shape of the curves is due to a downsampling of the simulation result. These initial values exclude either a pure Toda lattice or a purely linear ODE. This picture reveals the typical behaviour of soliton solutions: their shape is constant over time and is not changed by head-on collisions.

6 Conclusion

In this paper we have shown how two famous nonlinear lattice equations, the FPU lattice and the Toda lattice can be mapped onto a CNN. For this purpose certain original restrictions on the templates, such as symmetry, were dropped. Nevertheless the system is still stable, because the underlying mechanical models are stable. The mapping is exact under certain conditions for the state variables or initial conditions, respectively. Otherwise, the nonlinear output map of

[1]Strictly speaking they are only stable.

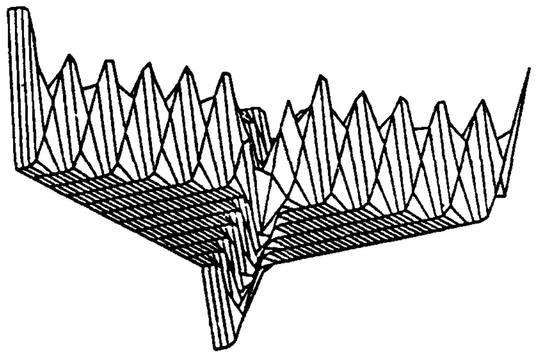


Figure 3: Soliton solution in modified Toda lattice with head on collision of two solitons

the CNN modifies the lattice equations. This was demonstrated for the Toda lattice. Numerical experiments and some analytical reasonings show that the sorting operations can be performed for certain initial conditions. It has to be stressed that the system is not Hamiltonian, but possesses symmetries in time.

References

[1] E. Fermi, J. R. Pasta, and S. M. Ulam, "Studies of nonlinear phenomena, 1965, Los Alamos report la 1940, May 1955," in *Collected Works of E. Fermi, Vol.2*, pp. 978–988, Univ. of Chicago Press, 1965.

[2] M. Toda, "Studies of a non-linear lattice," *Phys. Rep.*, vol. 8, pp. 1–125, 1975.

[3] W. W. Symes, "The QR algorithm and scattering for the finite nonperiodic toda lattice," *Physica 4D*, pp. 275–280, 1982.

[4] R. W. Brockett, "Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems," *Lin. Algebra & Applic.*, vol. 146, pp. 79–91, 1991.

[5] S. Paul and K. Hüper, "Analog rank filtering," Tech. Rep. TUM-LNS-TR-91-21, Technical University Munich, November 1991.

[6] S. Paul, K. Hüper, and J. Nossek, "A simple analog rank filter," in *Proc. IEEE Int. Symp. on Circuit and Systems*, 1992.

[7] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 1257–1272, Oct. 1988.

[8] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 1273–1290, Oct. 1988.

[9] T. Roska, C. W. Wu, M. Balsi, and L. O. Chua, "Stability and dynamics of delay-type and nonlinear cellular neural networks," Tech. Rep. UCB/ERL M91/110, Electronics Research Laboratory, Univ. California, Berkeley, 1991.

[10] H. Flaschka, "On the Toda Lattice I," *Phys. Rev. B*, vol. 9, pp. 1924–1925, 1974.

[11] H. Flaschka, "On the Toda Lattice II," *Prog. of Theor. Physics*, vol. 51, pp. 703–716, 1974.

[12] M. J. Ablowitz and P. A. Clarkson, *Solitons, Nonlinear Evolution Equations and Inverse Scattering*. Cambridge: Cambridge University Press, 1991.

[13] M. Toda, "Nonlinear lattice and soliton theory," *IEEE Trans. Circuits Syst.*, vol. 30, pp. 542–554, 1983.

SOME NOVEL CAPABILITIES OF CNN: GAME OF LIFE AND EXAMPLES OF MULTIPATH ALGORITHMS

Leon O.CHUA+, Tamás ROSKA, Péter L.VENETIANER and Ákos ZARÁNDY
Dual and Neural Computing Systems Laboratory
Computer & Automation Institute (MTA SzTAKI) Research Division, Hungarian Academy of Sciences
Kende u. 13, H-1111 Budapest, Hungary; H-1518 Bp, POB 63
+Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California at Berkeley, Berkeley, CA 94720

ABSTRACT

It is shown that the game of life algorithm, which is equivalent to Turing machine, can be realized by CNN. Next, a multipath CNN algotrithm is shown demonstrating the capabilities of analog/logic (analogic) software.

1. INTRODUCTION AND THE MAIN RESULTS

It is well known that in the realm of logic computing Turing machines are universal in a sense that any conceivable algorithm (recursive function) can be realized with it. In the field of analog regular processing arrays it has been shown that any nonlinear operator with fading memory can be realized by at most 4 layers of neural networks containing memoryless nonlinearities and delays [2]. Cellular neural network (CNN) is a new paradigm [1] for locally connected, geometrically placed, analog, 3D regular processing arrays. By introducing the nonlinear and delay-type templates and additional capabilities [3] we have an extremely broad universe of functionalities. Multi-layer perceptrons can be realized also by a class of cellular neural networks [4]. It can be shown that all the three types of partial differential equations can be approximated by CNNs [3c]. The programmable CNNs [3] provide a new quality: a simple way to solve CNN algorithms, a kind of analog software.

The silicon realization of CNN is convincing: the first tested CNN array has a capability of 0.3 TeraXPS on a chip [9] with 2 micron technology (fixed template) and the first programmable dualy computing CNN chip provides the full programmability [8].

Hence, it seems that the programmable CNN, in the above sense, is an universal programmable analog array computer.

In this report we show that

- the game of life can be realized by CNNs which means that any Turing machine can be realized by simple programmable CNNs,
- complex image processing tasks can be realized by programmable dual computing CNN by showing three useful CNN solutions (one shown in details)

Thus, we can prove that the programmable CNN is, theoretically, a universal machine in both the analog and the logical field. Moreover, for a broad class of problems the practical CNN algorithms represent a highly efficient truly parallel solution. In view of recent neurobiological results [10,5,6,15] and our first biological CNN models [11] it seems almost certain that the programmable CNN provides a very strong modelling paradigm for a lot of living systems.

2. THE GAME OF LIFE TEMPLATES

2.1. The basic step template

Being a no-player game, the life game [7] is not really a game in its traditional meaning. In principle it is "played" on an infinite squared board (cell array). At any time some of the cells will be **alive** and the others **dead**. After setting up an initial table, in each time step, the next state is defined by the following rules [7]:

BIRTH: a cell that is dead at time t becomes alive at time t+1 only if *exactly 3* of its eight neighbors were alive at time t.

DEATH: a cell that is alive at time t and has *less than 2 or more than 3* alive neighbors at time t will be dead by time t+1.

SURVIVAL: a cell that was alive at time t will remain alive at t+1 if and only if it had *exactly 2 or 3* alive neighbors at time t

The facts, that the cells of the array can be projected onto those of the CNN and the new state of a cell is determined only by the previous state of it and that of its neighbors, just like with the CNN, are appealing to realize the game with a CNN. The 2 layer network of Fig. 1. makes one step of the game.

$$A_{11}=[\ 1\ ] \qquad A_{12}=[\ a\ ] \qquad A_{22}=[\ 1\ ] \qquad I_1=1 \qquad I_2=-0.8$$

$$B_{11}=\begin{bmatrix} 0.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix} \qquad B_{21}=\begin{bmatrix} -0.6 & -0.6 & -0.6 \\ -0.6 & 0 & -0.6 \\ -0.6 & -0.6 & -0.6 \end{bmatrix}$$

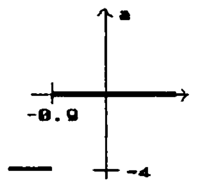


Fig. 1: The 2 layer single-step game of life template. The initial state has to be fed on the input of layer 1, and the new state will appear on the output of layer 2.

To understand how this network works, let's reformulate the rules of the game: a cell will be alive if *at least 3 of the 9 cells in its 3x3 neighborhood are alive* AND *at most 3 of its 8 neighbors are alive*. Layer one implements the first part of this conjunction, layer 2 realizes the second part and the conjunction itself (see Fig. 2.).

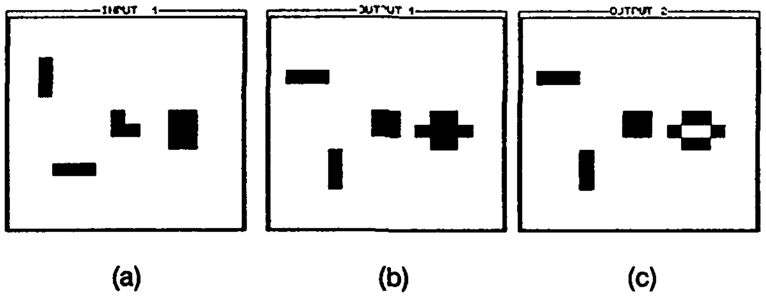


Fig. 2: The single-step life algorithm. a: Input pattern, b: output of layer 1, c: new state (output of layer 2)

2.2 Versions of a multistep CNN algorithm

2.2.1 Discrete Time CNN with thresholding sigmoid

The algorithm described in the previous chapter executes a single step of the life game with one transient. It would be much more useful, if we could simulate a whole life history with one CNN, i.e. the transient would settle down only if the game reached a stable state, otherwise it would run forever (see [7] for some stable and oscillating patterns and the famous forever growing glider gun). To reach our goal, two transients should be composed somehow: one transient between two states and another from the initial state to a final state or to eternity. For this reason the output should be fed back to the input, inducing a new transient. But there is a problem with timing: a new step can begin only after the termination of the previous one. Unfortunately, simply feeding back layer 2 of the above described CNN to the input does not meet this condition, because the time required to reach the new state (all cells have values ±1) depends on the number of neighbors, i.e. it alters from cell to cell, causing a terrible confusion. This problem can be solved most easily with a Discrete Time Cellular Neural Network (DTCNN) [12], which is equivalent to the CNN calculation using a forward Euler iteration with stepsize 1 and thresholding *sigmoid* [13]. Here the output state of a cell is binary (±1) and is determined by the sign of the state of the cell. The above described algorithm needs to be slightly modified: three layers are required, one for each part of the conjunction and a third layer to realize the conjunction itself, yielding the output pattern. (Fig. 3.).

$$A_{31} = [\ 1\ ] \qquad A_{13} = \begin{bmatrix} 0.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix} \qquad I_1 = 1$$

$$I_2 = -0.8$$

$$A_{32} = [\ 1\ ] \qquad A_{23} = \begin{bmatrix} -0.6 & -0.6 & -0.6 \\ -0.6 & 0 & -0.6 \\ -0.6 & -0.6 & -0.6 \end{bmatrix} \qquad I_3 = -1.5$$

Fig. 3: The 3 layer DTCNN template executing the whole life history.

2.2.2 Discrete Time CNN with piecewise-linear thresholding

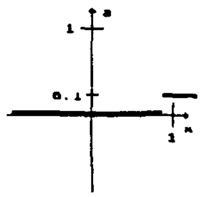
Another solution can be achieved with a DTCNN using piecewise-linear thresholding, just like in the continuous-time CNN. Using this network the output can take any value between -1 and +1. These values can be used to code information about the appropriate cell. Using nonlinear templates [3a,b] we can extract the coded information. The template of Fig. 4. realizes the multistep game of life algorithm in one layer. The first step of the algorithm is to code the information about the *previous state* and the *number of alive neighbors* of the current cell, and the second step is to decode this information, i.e. to drive the pixel black or white, depending on the coded value. This means that a new life pattern appears after every second iteration step. The result of the first, coding step is:

*base number + 0.1 * number of alive neighbors*

where the base number is -0.5 and -0.45 for the white (-1) and black (+1) cells, respectively. *Function b* "counts" the base number and *function a* the number of alive neighbors at $x=\pm 1$.

Consequently, the second step, the decoding phase should drive the pixel black, if the output of the first step is in the [-0.25,-0.15] domain. This is where the value of *function b* is +1.

$$A = \begin{bmatrix} a & a & a \\ a & b & a \\ a & a & a \end{bmatrix} \qquad \begin{matrix} B=0 \\ I=0 \end{matrix}$$



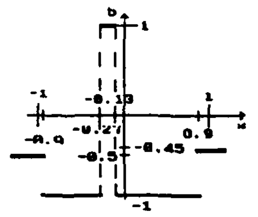


Fig. 4: Multistep game of life template for piecewise-linear DTCNN

3. MULTIPATH CNN ALGORITHMS

Due to local interconnectivity the CNN seems to be mainly suitable for detecting local properties, however some global properties can be extracted as well. Moreover, using the programable CNN multipath algorithms even more complex problems can be solved.

3.1 Blob-counting

Our task is to count blobs in a grey-scale image. The key of the algorithm is to associate a blob with a well-defined significant point and after suppressing the other pixels we only have to count these significant points. Shifting the remaining pixels to one side of the picture (e.g. with horizontal CCD [14]) the counting of them becomes easy.

<u>The basic idea of the algorithm</u>

Let's associate each blob with its local southern places (i.e. with pixels having neither south-eastern, nor southern, nor south-western direct neighbors). In two cases a blob can have more than one southern place:

(i) The blob has a horizontal southern edge (Fig. 5e);

(ii) The blob is concave from the bottom (Fig. 5b,5d and their vertical mirrors);

But fortunately neither causes any problem, because in case (i) the horizontal CCD contracts the southern places into a single pixel, and in case (ii) each concave place yields one new local southern place, so the difference of the number of the local concave places and the southern places gives the correct result, ie. the number of the blobs.

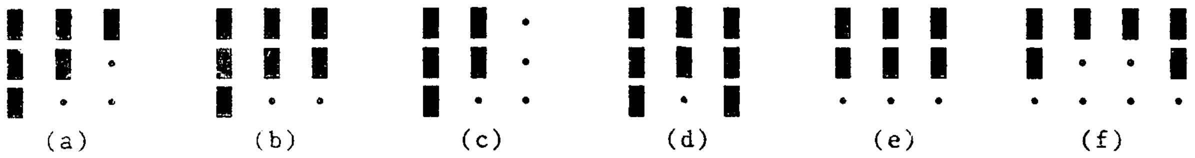


Fig. 5: In 3X3 neighborhood these bottom edges (and their vertical mirrors) could occur (a,b,c,d,e). The LCP template extracts two pixels from location (f). The ▮ sign denotes the blob pixels, and the • sign denotes the background pixels.

Holes can cause problems, because their local northern point is local concave place, but it doesn't increase the number of the local southern places. So if we want to know the number of the blobs we have to fill their holes first. Unless we do it, the algorithm gives the Euler number of the image, i.e. the difference of the number of blobs and the number of holes.

The templates and the flow diagram of the algorithm

The complete algorithm is shown on Fig. 7. The templates of the algorithm work on black-and-white pictures, so the first step is a grey-scale to black-and-white transformation using the average template [1]. In the second step we have to fill the holes with the hole-filler template [14]. We call the output of these transformations the preprocessed picture.

At this point the algorithm branches into two independent parallel paths. In the first branch the Local Southern Elements (LSE) are extracted by the LSE template (Fig. 6a). Next they are shifted horizontally with the CCD transformation. The last step of this branch is to count the black pixels. It is easy using a logical algorithm because they are only on one side of the picture in determined order. The CCD transformation causes in this case a large data reduction without any information losses.

In the second branch the task is very similar. The only difference is that instead of the LSE template, the Local Concave Places detector (LCP template Fig. 6b) is used. The LCP template associates one pixel with each local concave place except in the situation depicted on Fig. 5f, where two pixels are extracted but the horizontal CCD transformation melts these two neighboring pixels into one.

Finally, the last step is the subtracting the number of LSPs from the number of LSEs.

$$A=[\ 2\ ] \quad B=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad I=-5.5 \qquad A=[\ 2\ ] \quad B=\begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ -0.5 & -1 & -0.5 \end{bmatrix} \quad I=-5.5$$

(a) (b)

Fig. 6: The LSE (a) and the LCP (b) templates

An example: Counting keys

A grey-scale camera picture was taken of seven keys with a resolution of 63x62 pixels (Fig. 8a). First it was transformed to black-and-white (Fig. 8b), and then the holes were filled (Fig. 8c). Next, following the two parallel branches, the local southern elements (Fig. 8d: 24 connected components) and the local concave places (Fig. 8e: 17 connected components) were extracted. *Finally: 24-17=7: it's correct!*

In case of a conservative analog VLSI realization it could take about 20 microseconds (it depends only on the size of the picture).

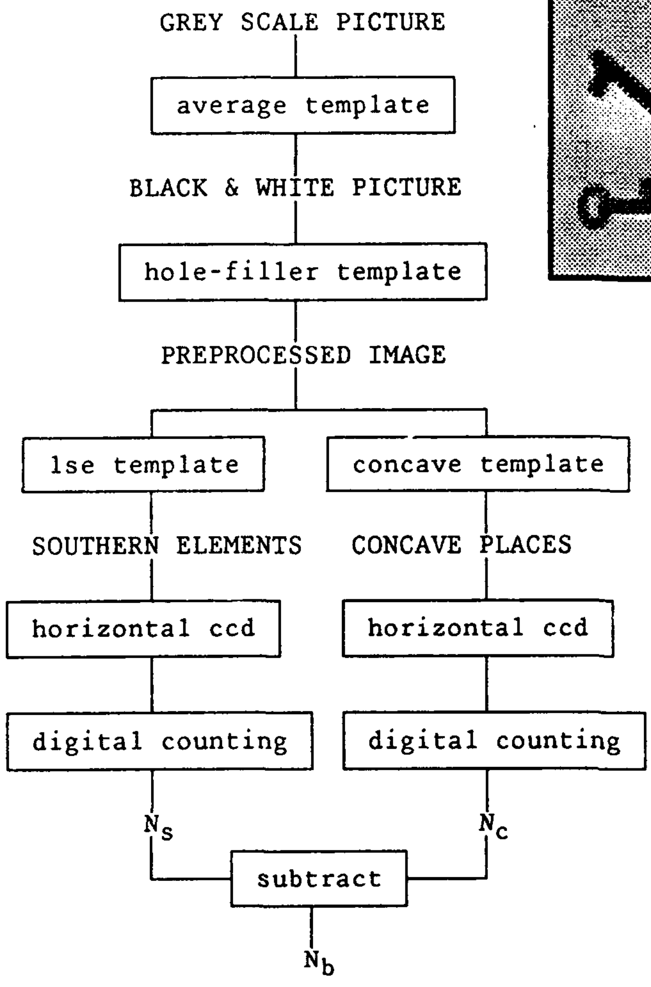


Fig. 7: The flow-diagram of the blob counting algorithm. Capital letters are used to denote images.

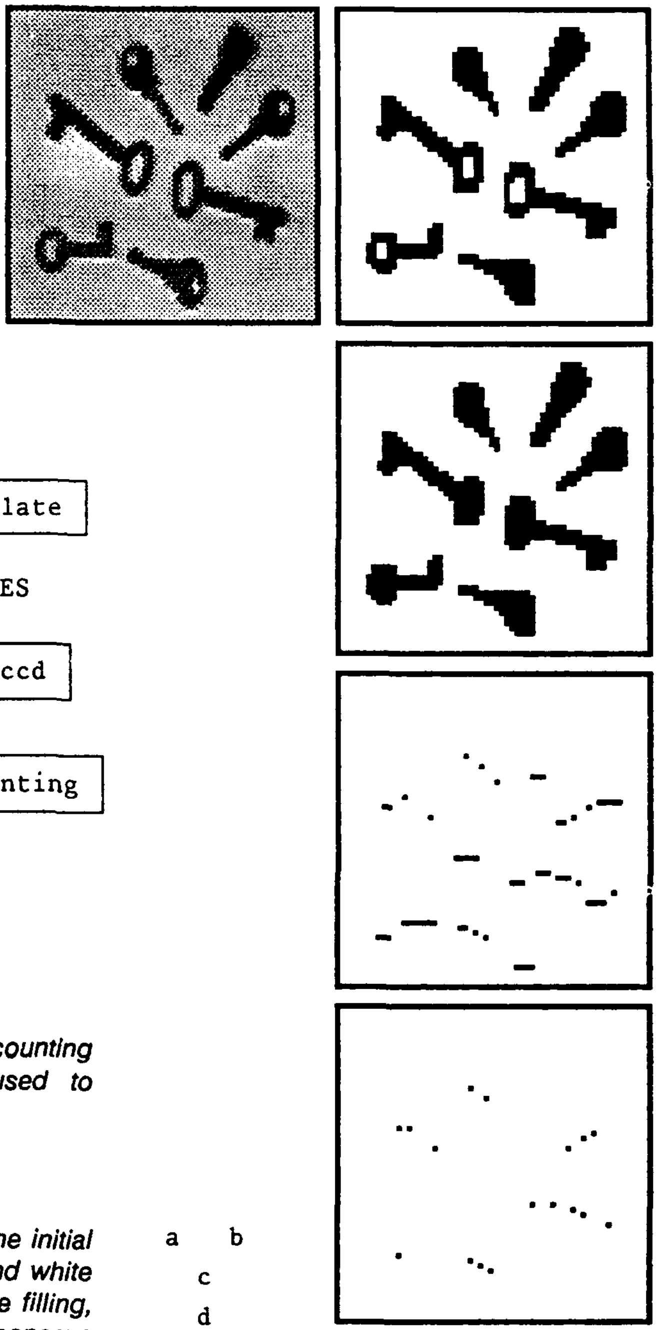


Fig. 8: The five steps of counting keys: the initial grey-scale image (a), the black and white image before (b) and after (c) hole filling, the local southern (d) and concave elements (e)

a b
c
d
e

3.2. Other multipath CNN algorithms

Two other multipath CNN algorithms were realized. The first one can make blob size classification. This probabilistic algorithm can classify larger blobs than the template size. The second one can detect textile-pattern errors. It works on a simple sleazy weaved textile, and can detect the fiber breakings and the knots.

ACKNOWLEDGEMENTS

This work has been supported by the research grant No. 2578/91 of the National Research Fund of Hungary (OTKA) and a joint grant No. INT 90-01336 of the National Science Foundation of the United States of America and the Hungarian Academy of Sciences.

REFERENCES

[1] a L.O.Chua and L.Yang, "Cellular neural networks:Theory", IEEE Transactions on Circuits and Systems, Vol.35, pp.1257- 1272, 1988.

b L.O.Chua and L.Yang, "Cellular neural networks: Applications", ibid., pp.1273-1290.

[2] T.Roska, "Analog events and a dual computing structure using analog and digital circuits and operators", pp.225-238 in Discrete Event Systems: Models and Applications (eds. P.Varaiya and A.B.Kurzhanski), Springer-Verlag, Berlin, 1988.

[3] a T.Roska and L.O.Chua, "Cellular neural networks with nonlinear and delay-type template elements", Proc. IEEE CNNA-90, pp.12-25, 1990

b T.Roska and L.O.Chua, "Cellular neural networks with nonlinear and delay-type template elements and non-uniform grids", Int.J.Circuit Theory and Applications, to appear

c T.Roska and L.O.Chua, "The dual computing analog software", Report DNS-2-1992, Dual and Neural Computing Systems Laboratory, Comp.Aut.Inst., Hungarian Academy of Sciences, Budapest, 1992

[4] J.A.Nossek and G.Seiler, "An equivalence between multi-layer perceptrons with stepfunction type nonlinearity and a class of cellular neural networks", Report No.:TUM-LNS-TR -90-7, Inst. Network Theory and Circuit Design, T.U.Munich, Munich December, 1990

[5] a J.Allman, F.Miezin, and E.McGuiness, "Stimulus specific responses beyond the classical receptive field: neurophysiological mechanisms for local-global comparisons in visual neurons", Ann.Rev.Neurosci., Vol.8, pp.407-430, 1985

b G.Montgomery, "The mind's eye", Discver, May 1991, pp.51-56

[6] W.Heiligenberg, "The neural basis of behavior: a neuroethological view", Ann.Rev.Neurosci.,Vol.14, pp.247-267, 1991

[7] E.Berlekamp, J.H.Conway, and R.K.Guy, Winning ways, Academic Press, New York, 1982, Chapter 25, What is life?, pp.817-850,

[8] K.Halonen, V.Porra, T.Roska and L.O.Chua, "VLSI implementation of a reconfigurable CNN containing local logic", Proc. CNNA-90, pp.206-215, 1990

[9] J.M.Cruz and L.O.Chua, "A CNN chip for connected component detection", IEEE Trans. Circuits and Systems, Vol.38, pp.812-817, 1991

[10] J.Hámori, T.Pasik, P.Pasik and J.Szentágothai, "Triadic synaptic arrangemetns and their possible significance in the lateral geniculate nucleus of the monkey", Brain Research, Vol. 80, pp. 379-393, 1974

[11] T.Roska, K.Lotz, J.Hámori, E.Lábos and J.Takács, "The CNN model in the visual pathway - Part I: The CNN-Retina and some direction- and length-selective mechanisms, Report DNS-8-1991, Dual and Neural Computing Systems Laboratory, Comp.Aut.Inst., Hungarian Academy of Sciences, Budapest, 1992

[12] H.Harrer and J.A.Nossek, "Time discrete cellular neural networks: architecture, applications and realization", Report No. TUM-LNS-TR-90-12, Technical University of Munich, November 1990, to appear in Int.J. Circuit Theory and Applications

[13] T.Roska, "On the qualitative and quantitative relationships between the analog and digital realizations of neural computing circuits", Report DNS-2-1989, Dual and Neural Computing Systems Laboratory, Comp.Aut.Inst., Hungarian Academy of Sciences, Budapest, 1989

[14] T.Matsumoto, T.Yokohama, H.Suzuki, R.Furukawa, A.Oshimoto, T.Shimmi, Y.Matsushita, T,Seo and L.O.Chua "Several Image Processing Examples by CNN" 1990 International Workshop on Cellular Neural Networks and their Aplication CNNA-90 Proceedings, pp 100-112,

[15] D.C.Van Essen, C.H.Anderson, and D.J.Felleman, "Information processing in the primate visual system: an integrated systems perspective", Science, Vol.255, pp.419-423, January 24, 1992

CNNA'92

The distinguished members of the Scientific Committee of the Second International Workshop on Cellular Neural Networks and their Application take pleasure in presenting to

P. Thiran **H. Harrer**

this Certificate in recognition of the scientific achieve- ɔn
ment attained as represented by the contributions

Two Causes of Instability of Cellular Neural Networks

New Test Results of a 4x4 DTCNN Chip

which has been selected as the Most Outstanding Submitted Paper of CNNA'92.

CNNA'92, Munich Germany
October 14–16, 1992

The Scientific Committee

Author Index

Author	Session	p.
Aizenberg , I.N.	Design	36
Aizenberg , N.	Design	36
Andreyev	Theory I	135
Balsi	Theory I	129
	Theory II	246
Barone	Theory II	246
Bito	Application	216
Belsky	Theory I	135
Boros	Application	216
Chigawa	Application	191
Chua	inaugural	1
	Learning	55
	Learning	68
	Theory I	106
	Theory II	276
Cimagalli	invited	92
	Theory II	246
Civalleri	Chairman	55
	Theory I	94
Crounse	Design	29
Dabrowski	Theory I	123
Dalla Betta	VLSI	151
Dehaene	Theory II	234
Dmitriev	Theory I	135
Erőss	Application	216
Espejo	VLSI	169
Frühauf	Theory I	106
Füssl	Learning	55
Galias	Design	23
Gilli	Theory I	94
Graffi	VLSI	151
Güzelis	Learning	74
Hansen	Learning	62
Harrer	VLSI	163
	Application	238
Hasler	Chairman	94
Heiligenberg	invited	-
Ho	Application	210
	Theory II	252
Huertas	invited	141
	VLSI	169
Jankowski	Application	197
Kacprzak	VLSI	157
Kaluzny	Theory I	112
Kispal	Application	203
Kiss	Application	216
Köhn	Application	238
Kovacs	VLSI	151
Kozek	Application	203
Kufudaki	Design	42
Kuminov	Theory I	135
Lüder	invited	45
	Theory I	106
Magnussen	Learning	80
Martinelli	Theory I	117
Masetti	VLSI	151
Mathis	Design	417
	Chairman	191
Mizutani	Learning	86
Mori	Application	210
	Theory II	252
Moschytz	Design	11
Nachbar	Learning	55
	Learning	68
Nakayama	Application	191
Nossek	Learning	55
	Learning	68
	Learning	80
	VLSI	163
	Application	238
	Theory II	258
	Theory II	270
Orgorzalek	Theory I	123
Oosterlinck	Theory II	264
Osuna	Design	11
Paul	Theory II	270
Perfetti	Theory I	117
Porra	Chairman	151
Price	Theory II	264
Radványi	Application	216
	Application	222
Ramacher	Chairman	234
Rodriguez-Vasquez	VLSI	169
Roska	Design	29
	invited	181
	Application	216
	Theory II	276
Savaci	Theory II	240

Schuler	Learning	55	Ueda	Theory II	252
	Learning	68	Vandewalle	Theory II	234
Schwarz	Design	17		Application	240
Seiler	Theory II	258	Vass	Application	216
Slot	VLSI	157	Venetianer	Theory II	276
	VLSI	175	Wallinga	Chairman	11
Smith	Application	238	Wambacq	Theory II	264
Szolgay	Application	203	Wanczuk	Application	197
Takahashi	Theory II	252	Yu, D.	Application	210
Tanaka	Design	29	Yu, X.	Application	210
Thiran	Theory I	100	Zarandy	Theory II	276