ATR's CAM-Brain Machine (CBM) Simulation Results and Representation Issues[1]

Norberto Eiji Nawa
Nagoya University & ATR - Human Information Processing Laboratories
Seika-cho, Kyoto, 619-0288, Japan
xnawa@hip.atr.co.jp
www.hip.atr.co.jp/~xnawa

Hugo de Garis
ATR - Human Information Processing Laboratories,
Seika-cho, Kyoto, 619-0288, Japan
degaris@hip.atr.co.jp
www.hip.atr.co.jp/~degaris

Felix Gers
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale,
Corso Elvezia 36, CH-6900 Lugano
Switzerland
felix@idsia.ch, www.idsia.ch/~felix

Michael Korkin
Genobyte, Inc.
1319 Spruce Street, Suite 210 Boulder
CO 80302 USA
korkin@genobyte.com, www.genobyte.com

ABSTRACT

This paper presents some simulation results of ATR's new CAM-Brain Machine (CBM), a piece of FPGA based hardware to be operational by the summer of 1998, which will update 3D cellular automata (CA) cells at the incredible rate of 100 Billion a second, making possible the evolution of a CA based neural net module in about a second (i.e. a complete run of a genetic algorithm, with tens of thousands of neural circuit growths and fitness evaluations). Since the ultimate objective of ATR's CAM-Brain Project is the realization of neural networks modules containing some billion artificial neurons to run in real time, speed is a critical issue. The main objective of the experiments presented in this paper was to gain insights and experience concerning the evolvability of the CoDi-1Bit model (a simplified version of the previous CoDi model that allows 1 bit neural signaling, thus enabling its implementation in FPGA based hardware) and the new CBM machine. As a subgoal, and no less important, we have investigated some representation issues, namely how to interpret the binary input and output signals of neural network modules. This issue is fundamental, since it strongly influences the evolvability and performance of the whole CAM-Brain system.

1 Introduction

The CAM-Brain Project at ATR Labs aims to construct a large-scale brain-like neural network system. If the project succeeds and our expectations are fulfilled, these "artificial brains" will have a large number of potential applications in several different fields, from 'smart' domestic appliances to speech processing and robot control. We believe that to realize a system that possesses a similar level of functionality and structural complexity as real biological brains, the most appropriate way, if not the only way, is to evolve them, as happened in nature.

The fundamental approach of the CAM-Brain Project is the growth/evolution of large-scale neural networks. Since the dawn of the Project (de Garis 1994), Cellular Automata (CA) have been chosen as the medium to grow the neural networks. CAs meet the requirements of generality and especially scalability, necessary for simulating large-scale systems. Moreover, the parallel nature of CAs allows its transposition into hardware, where higher speeds can be achieved. The CA based neural model initially used (de Garis 1996) suffered from an explosion of

[1] In Koza, J.R, Banzhaf, W, Chellapilla, Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H., and Riolo, R. (editors). *Genetic Programming 1998: Proceedings of the Third Annual Conference*, July 22-25, 1998. University of Wisconsin, Madison, Wisconsin, San Francisco, CA: Morgan Kaufmann

states and transition rules that blocked any attempt to implement it in hardware. Due to this problem, a new model called "CoDi" (from COllect and DIstribute) was proposed (Gers and de Garis 1996), greatly simplifying the system and for the first time allowing the implementation of the system in special hardware, namely XC6264 FPGAs, which update the CA space at 100 billion cells a second, and will be able to perform a complete run of a genetic algorithm (with 10,000s of circuit growths and evaluations) in about a second.

Advances in hardware technology led to the development of devices called Field Programmable Gate-Arrays (FPGAs). FPGAs are hardware devices that can be reconfigured in run-time to perform different logic functions, wedding the flexibility of software with the speeds of hardware. This motivated the design/construction of a specific computer, called CAM-Brain Machine (CBM) (Korkin *et al.* 1997), for the evolution of neural networks under the CoDi model. The CBM will grow 16.000 neural network modules of roughly 10.000 CA cells each, updating 100 billion cells/second, a speedup of 500 times compared to the MIT machine "CAM8" (Toffoli and Margolus 1987) that the Project had been using previously to update CA cells quickly.

The experiments reported in this paper had as their main objective to allow us to gain some insights and ideas concerning the evolvability of the CoDi-1Bit CA based neural net model and the CBM. The experiments were not exhaustively performed. The current simulations were executed in conventional workstations and laptops and often took days until convergence was achieved. The hardware version of the CBM will dramatically decrease these times, thus facilitating a more complete analysis. Also, we began tackling an issue of great importance, namely, how to interpret the signals that come from the neural networks modules? The CoDi-1Bit model evolves neural networks whose signals that traverse the connections are digital, i.e. binary 0's and 1's. The question is, what kind of representations should be used in order to extract useful information from the signals output by the neural networks modules. Simple representation schemes have been tried, with mixed results - some good, some bad. Also, we attempted to apply the theory presented in the book "Spikes: exploring the neural code" (Rieke *et al.* 1997), which introduces a novel hypothesis to explain how sensory signals are encoded in the action potentials that traverse natural neural systems. The initial results obtained are encouraging and indicate that a "spike interval coding" representation might be suitable to be used with the CoDi model.

2 The CoDi-1 Bit Cellular Automata Based Neural Net Model

This section gives an overview of the "CoDi"neural net model implemented in the CAM-Brain Machine hardware. The model is called "CoDi" due to the "COllect and DIstribute" nature of its neural signals. CoDi is a simplified CA-based neural network model developed at ATR in the summer of 1996 with two goals in mind. One was to make neural network functioning much simpler compared to the older CAM-Brain model developed in 1993 and 1994 (de Garis 1993, de Garis 1994), so as to be able to implement the model directly in electronics and thus to evolve neural net modules at electronic speeds.

In order to evolve one neural network module, a population of modules is run through a genetic algorithm for several hundred generations. Each module evaluation consists of growing a new set of axonic and dendritic trees which interconnect the neurons in the 3D cellular automata space, then running the module to evaluate its performance (fitness).

The CoDi model (Gers and de Garis 1996) operates as a 3D cellular automata. Each cell is a cube which has six neighbor cells, one for each of its faces. By loading a different phenotype code into a cell, it can be reconfigured as a neuron, an axon, or a dendrite. Neurons are configurable on a coarser grid, namely one per block of 2*2*3 CA cells. In a neuron cell, five (of its six) connections are dendritic inputs, and one is an axonic output. An accumulator sums incoming signals and fires an output signal when a threshold is exceeded. Each of the inputs can perform an inhibitory or an excitatory function (depending on the neuron's chromosome) and either adds to or subtracts from the accumulator. The neuron cell's output (axon) can be oriented in 6 different ways in the 3D space.

A dendrite cell also has maximum five inputs and one output, to COllect signals from other cells. The incoming signals are passed to the output according to a given function. For instance, if the logic OR function is used, the output is active whenever at least one of the inputs is active. If an XOR function is used, the output is active when only a single input is active. Two or more active inputs block each other. The XOR dendrite is more plausible from the biological point of view. A similar phenomenon occurs in real dendrites in animals. An axon cell is the opposite of a dendrite. It has 1 input and maximum 5 outputs, and DIstributes signals to its neighbors.

Before the growth begins, the module space consists of blank cells, which are used to grow new sets of dendritic and axonic trees during the growth phase. Blank cells perform no function in an evolved neural network.

As the growth starts, each neuron continuously sends growth signals to the surrounding blank cells, alternating between "grow dendrite" (sent to the neuron's dendritic connections) and "grow axon" (sent to the axonic connection). A blank cell which receives a growth signal becomes a dendrite cell, or an axon cell, and further propagates the growth signal, being continuously sent by a neuron, to other blank cells. The direction of the propagation is guided by the growth instructions attached to the cell. These local instructions indicate the directions that the growth signal should be propagated to and consists of a bit for each face of the cube cell. The growth signal is propagated to those directions whose corresponding bit is set to 1 (except the direction where the signal comes from).

This mechanism allows the growth of a complex 3D system of branching dendritic and axonic trees, with each tree having one neuron cell associated with it. The trees can conduct signals between the neurons to perform complex spatio-temporal functions. The end-product of the growth phase is a phenotype bitstring which encodes the type and spatial orientation of each cell.

3 CAM-Brain Machine (CBM)

The CAM-Brain Machine (CBM) was especially designed to support the growth and signaling of neural networks built by the CoDi model in hardware. The CBM should fulfill the needs for high speeds, when simulating large-scale binary neural networks, a necessary condition when one is concerned with performing real-time control. The hardware core is implemented in Xilinx's XC6264 FPGA chips, in which the neural networks will actually grow. A host machine will provide the necessary interface to interact with the hardware core. It is planned that the CBM will be used to grow 16,000 neural network modules, each with approximately 10,000 cells. The modules will be organized into humanly defined architectures, so that neural network modules will be interconnected to form a functional unity. This machine should be built by late summer of 1998, provided that the chips are delivered on time. For a complete description of the CBM, refer to (Korkin *et al.* 1997).

4 Some CBM Simulation Results

This section presents some CBM software simulation results using simple representation schemes. When one begins evolving CoDi modules, one immediately becomes conscious of the issue of representation. In other words, what meanings should one give to the binary inputs and outputs? We have considered 4 different representations. The first representation we tried was to consider two output streams tapped from different points, where each time one stream gave a pulse, a counter was incremented, and each time the other stream gave a pulse, the counter was decremented. This representation scheme was named "Incrementor/Decrementor" and we evolved a sinusoidal wave this way (See Fig.1). The second representation we tried was what we call "unary", i.e. if there are N input/output neurons that are firing at a given instant, then the module is said to be in/outputting the number "N" at that instant. We spent some time on the unary representation, because we thought it might be a good candidate for the CBM. Its advantage is that a different number can be in/output at each clock tick, i.e. it is a "fast" representation, which is important when building artificial brains containing long chains of sequential modules, so as to minimize total reaction time. The third representation we tried was "Gray code", which did not work very well and was abandoned. Finally, we attempted to apply the theory of the so called "spike interval coding", in which information is contained in the spacing between pulses. A pulse train coming from a single output neuron can be convoluted with an analog "filter function", to give a highly efficient means to code for complex time dependent analog output (Rieke *et al.* 1997). In the context of the CBM, the idea is to convolute an output series of 0-1 bits with a digitized filter in order to get the output signal. The next four subsections describe the experiments and the results.

4.1 Incrementor/Decrementor Representation

In this experiment we used a 16*16*16 CA block, with 6 fixed inputs constantly firing at the input face, and 2 fixed position outputs at the opposite face. If the first output bit B_1 was a 1 and the second bit B_2 was a 0, then a fictitious counter was incremented by 1. If $B_1 = 0$ and $B_2 = 1$, then the counter was decremented by 1. Other cases left the counter unchanged. An arbitrary sinusoid amplitude and wavelength target shape was chosen that the counter is supposed to follow as closely as possible, with the counter value being plotted along the vertical y axis, and the clock count being plotted along the horizontal x axis. The circuit should evolve so that the outputs B_1 and B_2 change over time, such that the counter gives the appropriate y values over time.

The fitness value was defined to be the inverse of the sum of the squares of the differences between the desired y values and the actual counter values. An incremental or stepwise evolutionary approach was taken that de Garis developed in 1990 (de Garis 1990), namely that the evolution used several intermediate fitness definitions, where the resulting population of GA chromosomes evolved with fitness definition FD_1, became the starting generation of chromosomes with fitness definition FD_2, etc. Often this approach gives better results than using only one fitness definition. In the case of the sinusoid, at first a quarter wavelength was evolved (where the target sinusoid had an amplitude of 20 and

a wavelength of 200 clock ticks), with the whole curve shifted up by 30. The fitness measurement started after 20 clocks, at which point, the counter was set initially at the value 30. After about 100 generations, the evolved output followed the desired target within a few percent. The resulting population became the starting population for a second evolutionary phase where the aim was to evolve a half sinusoid. Again, after a day, and several hundred generations, the actual curve followed the target curve within a few percent. Two further steps were used, to evolve a three-quarter wave and a full wave. The three-quarter wave evolved as well as the previous two, but the full wave did not quite get the actual curve to return to the "base line" (set at 30). It looked as though the limits of the evolvability of the 4K CA cell module had been reached. An alternative fitness definition which weighted the later clocks was attempted, but had no effect. Fig. 1 shows the result of the full sinusoid, both target and actual output.

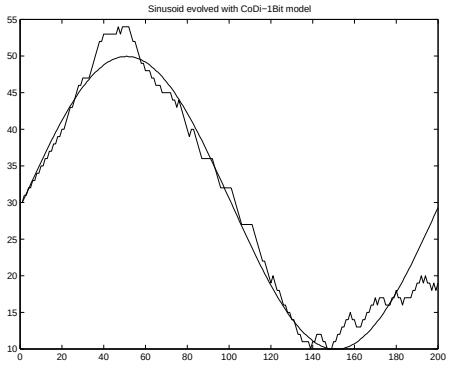

Figure 1 Full Sinusoid, incrementor/decrementor approach.

4.2 Unary Representation

The unary representation was expected to be less evolvable than a fixed output position representation. For example, imagine we input 30 pulses distributed randomly over the input surface, and that at each clock tick, the random positions of the 30 input pulses can change. (This is needed in case the unary output of one module feeds directly into the correspondingly positioned inputs of another unary module). The output positions at the opposite face can also change randomly (only their total number matters). Thus the unary representation is highly stochastic. Could, nevertheless, a CoDi-1Bit module evolve a desired time dependent unary output? Fig. 2 shows such a case. The straight line is the target output. The spiky line is the actual output (unary) over time. One can see that the unary representation is not bad, but not perfect. It may be adequate for CAM-Brain modules if accuracies do not need to be finer than say within 10-20% error. Fig. 3 shows an experiment which aimed to achieve the greatest total output count. The greatest density of neurons in the CBM is one per 2*2 surface CA cells, so with a module of dimensions 24*24*16, means a maximum possible output of 12*12 = 144. Fig. 3 shows that the average maximum output was about 108 and the average deviation about this mean value was only about 3, i.e. a noise to signal ratio of about 3%, which was considered to be quite good. We then attempted to evolve a unary based sine curve, as shown in Fig. 4 which was a bit scratchy, due to the inevitable stochastic nature of the unary representation.

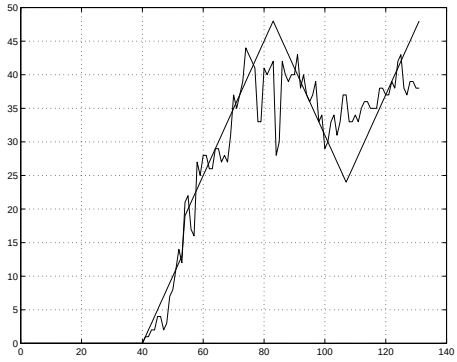

Figure 2 Obtained line and target line using unary representation.

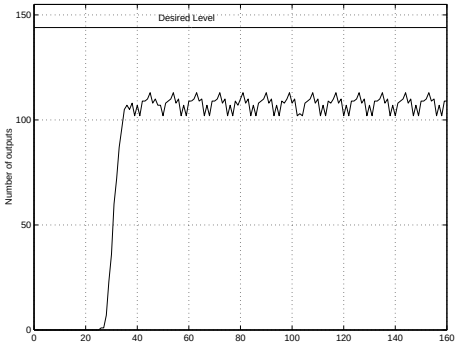

Figure 3 Maximum output experiment

The dotted line in Fig.4 represents the desired curve and the continuous line is the output. The evolution was performed incrementally. First, the first quarter of the sine, then the first and second, etc. as in the incrementor/decrementor experiment. Approximately 1000 generations were spent in each phase of the process.

Encouraged by the above moderate successes using unary representation, some attempts were made to evolve modules with controllable functions, e.g. an inverter. Using the same neural circuit, a unary input of 20 was desired to result in a unary output of 60 and vice versa. This circuit failed to evolve. Once the output reached about 20 it just hovered there. Instead of 20→60 and 60→20, a milder form, 8→16 and 16→8 was tried. This time the average output hovered around 12. We tried evolving a damper switch. The input surface was divided into two halves. In the right hand side, a unary input of 16 was placed. In the left hand side, all 72 possible input neurons were all switched on constantly, or

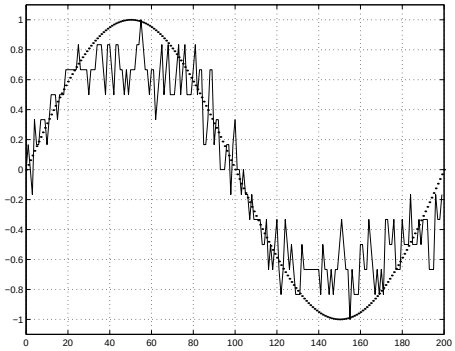

Figure 4 Example of a sinusoidal signal output by a neural network using the unary code.

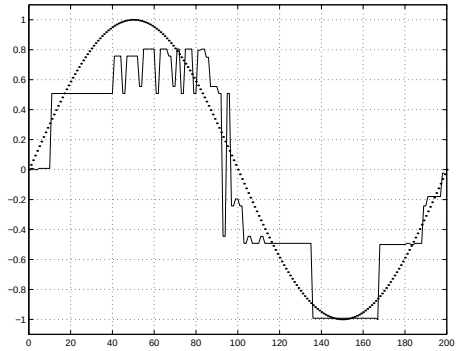

Figure 5 Example of a sinusoidal signal output by a neural network using the 8-bit Gray code.

all switched off constantly. If the switch was on (i.e. all 72 input neurons were firing) the output was supposed to be 8. If the switch was off (all 72 input neurons were off), the output was supposed to be 16. The output for both cases (switch on and off) hovered around 11. In the next experiment, the control (switch) input neurons were distributed evenly around the unary input points (to promote greater entanglement) with requirements - if switch is on, 16→3, if switch is off, 16→6. The result hovered around 5. These failures reduced our enthusiasm for the unary representation.

4.3 8-bit Gray code

We tried evolving a sine curve using an 8-bit Gray code. The reason why a Gray code was used instead of a simple binary code is that the Gray code of adjacent integers differ by only 1 bit. The most remarkable difference between the unary code and the Gray code is that the former has no weights for the output neurons/bits that comprise the word, increasing the level of redundancy in the set of representations (i.e. different representations lead to the same integer, as long the sum of the active output neurons is the same). On the other hand, the Gray code is more brittle and has no redundancy. The advantage of the Gray code is that it is more efficient in terms of usage of output neurons. When scaling up the range of the addressable values, the unary code linearly scales up the number of output neurons, the Gray code scales up logarithmically. Figure 5 shows an example of a sinusoidal obtained when using the 8-bit Gray code. We thought that it was not very successful and subsequently abandoned using it.

4.4 Spike Interval Coding

We attempted using the "spike interval coding" theory presented in (Rieke *et al.* 1997) to evolve sinusoidal waves, as in the previous experiments.

The procedure for decoding a spike train (the sequence of 0-1 bits) consists of convoluting it with a special "convolution filter" (See Fig.6). The result obtained is called the **estimated signal**, which is a time-dependent signal that is output from the neural network module to be evaluated in a fitness calculation or to be used in another process. The convolution process is discrete, since the convolution filter, the spike trains and the results are discrete. The estimated signal is a digital representation of an analog signal, sampled at discrete time points, corresponding to the clock ticks.

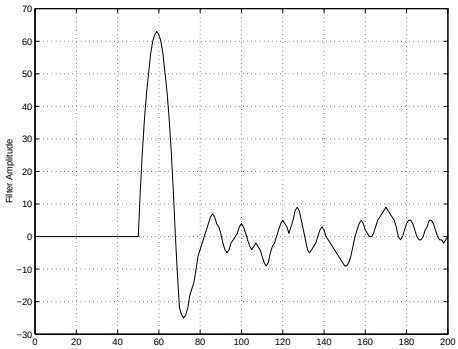

Figure 6 Decoding filter for the spike trains

The convolution process was tried on a single output of the CoDi neural network modules. The objective was to minimize the error between the estimated signal and a sine wave. The solid line in Fig.7 is the target sine wave, and the dashed line is the obtained wave after 600 generations. The output stream collected from the neural network modules had a length equal to 300 bits. The filter used in the convolution process a had length of 150 bits, and the length of the estimated signal was 120 bits. The population had 30 chromosomes and the CA space size was 24*24*18. 48 input points (neurons) were chosen from one of the faces of the cubic CA space and were constantly firing in order to bring a high level of activity to the module. The output signal was collected from a point in the opposite face of the inputs. The estimated signal was normalized, so its discrete time points would have values of the same order as a unitary sine wave. The obtained output signals are shown in Figures 7 to 10.

These results are much more encouraging than those of the other representations. Clearly, the "spike interval

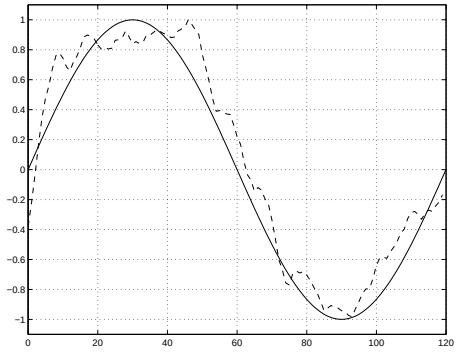

Figure 7 Single period of a sinusoidal wave generated using the "spike interval coding".

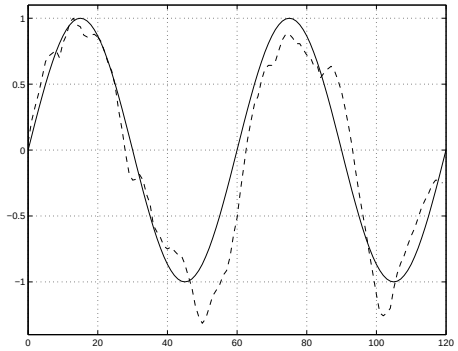

Figure 8 Two periods of a sinusoidal wave generated using the "spike interval coding".

coding" representation seems more suitable for the CoDi model. Further investigations in this direction will be the theme for future work.

5 Conclusions and Future Work

We presented some results using four different representation schemes. We felt the best results were obtained with the 4^{th} "spike interval coding". The CoDi-1Bit model appears to have rather good evolvability when it comes to evolving spike trains from a fixed output point.

Finally, the source code for the CoDi-1Bit neural net model can be downloaded from de Garis's web site at

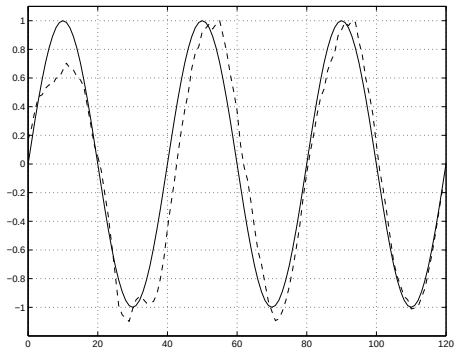

Figure 9 Three periods of a sinusoidal wave generated using the "spike interval coding".

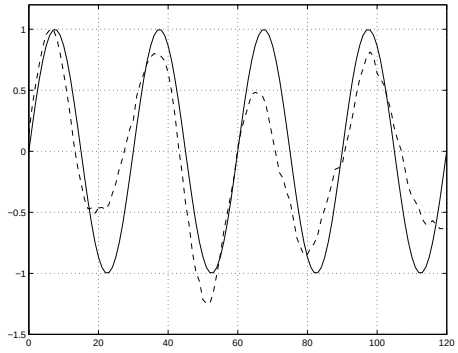

Figure 10 Four periods of a sinusoidal wave generated using the "spike interval coding".

http://www.hip.atr.co.jp/~degaris. Hopefully other researchers will be inspired by this paper to face the challenge of successfully evolving CoDi (or other model) based neural modules. If an artificial brain of a million modules is to be built, then thousands of human evolutionary engineers (EEs) will be needed. The CAM-Brain team welcomes collaborators.

Acknowledgements

The first author is grateful for the invaluable support of Katsunori Shimohara and Takeshi Furuhashi.

References

de Garis, Hugo (1990). Genetic programming: Building artificial nervous systems using genetically programmed neural network modules. In: *Proceedings 7th. Int.Conf.on Machine Learning* (B.W. Porter and R.J. Mooney, Eds.).

de Garis, Hugo (1993). Artificial life: Growing an artificial brain with a million neural net modules inside a trillion cell cellular automata machine. In: *4th. Int.Symposium on Micro Machine and Human Science.*

de Garis, Hugo (1994). An artificial brain : Atr's cambrain project aims to build/evolve an artificial brain with a million neural net modules inside a trillion cell cellular automata machine. *New Generation Computing Journal.*

de Garis, Hugo (1996). CAM-Brain: ATR's billion neuron artificial brain project: A three year progress report. In: *Proceedings of AROB'96, Int.Conf.on Artificial Life and Robotics.*

Gers, Felix and Hugo de Garis (1996). CAM-Brain: A new model for ATR's cellular automata based artificial brain project. In: *Proceedings of ICES'96, Int.Conf.on Evolvable Systems.*

Korkin, Michael, Hugo de Garis, Felix Gers and Hitoshi Hemmi (1997). CBM (CAM-Brain Machine):

A hardware tool which evolves a neural net module in a fraction of a second and runs a million neuron artificial brain in real time. In: *Genetic Programming 1997: Proceedings of the Second Annual Conference* (John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba and Rick L. Riolo, Eds.).

Rieke, Fred, David Warland, Rob de Ruyter van Steveninck and William Bialek (1997). *Spikes: exploring the neural code*. MIT Press/Bradford Books. Cambridge, MA.

Toffoli, T. and N. Margolus (1987). *Cellular Automata Machines*. MIT Press. Cambridge, MA.