DIGITAL NEURAL NETWORKS IN THE CAM-BRAIN MACHINE

H. EECKHAUT AND J. VAN CAMPENHOUT
Ghent University (RUG)
Department of Electronics and Information Systems (ELIS)
Parallel Information System (PARIS)
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
tel: +32 (0)9 2648965, fax: +32 (0)9 2643594
E-Mail: Hendrik.Eeckhaut@elis.rug.ac.be

This paper presents the underlying ideas, architecture and precise operation of the CAM-Brain Machine (CBM). The CBM is a hardware implementation of a brain-inspired, recurrent, digital, artificial neural network based on cellular automata. This note also reports on the results of the first experiments, performed on a physical CBM.
Keywords: CAM-Brain Machine (CBM), digital recurrent artificial neural networks, evolvable hardware.

1 Introduction

A known problem with artificial neural networks is the absence of a direct massively parallel implementation. To use neural networks, we still predominantly have to resort to simulations on traditional von Neumann architectures which rules out the massive exploitation of the inherent parallelism. This is a major drawback, in particular during the time-consuming training phase.

Recently, Genobyte Inc. (Boulder, Colorado, USA) built the CAM[a]-brain machine (CBM). This is an experimental machine composed of reconfigurable[b] (evolvable) hardware, capable of training and evaluating cellular automata based neural network modules directly in silicon. In the development of the CBM, the main objective was to design a computationally powerful evolvable hardware research tool for the evolution of complex digital circuits.[1] Considerable attention was devoted to implement the most time-consuming functions directly in hardware, so as to ensure the shortest possible execution time. Genobyte claims the CBM to deliver results unattainable or unpractical with any other technology.

It is expected that the CBM will indeed be effective in those applications where already today neural networks or cellular automata[2] are being used. Due to

[a] Cellular Automata Machine
[b] FPGA-based

Figure 1. The CAM-Brain Machine.

its hardware implementation, the CBM promises a (much) faster training and operation.
Although the machine is ingeniously constructed, researchers have not yet been able to develop real applications where the CBM outperforms traditional architectures. Even though the CBM is based on insights of the research field of artificial neural networks, it is not possible to naively take over the existing results and experience. CBM's paradigm strongly deviates from the traditional neural architectures because of its peculiar digital neural networks model. This model was especially designed for a direct implementation in electronic hardware; it had to be simple, compact and completely digital.
In comparison to 'analog' networks, relatively little is known about the performance and, in particular, the training of digital neural networks.
Why then would one want to implement completely digital networks? The reasons are plenty: in the first place for their flexibility and reliability. In addition, they avoid the disadvantages of analog technology[3]: susceptibility to noise and process-parameter variations (sensitivity to the production process) which limit computational precision and make it harder to understand what exactly is computed.

2 The CAM-Brain Machine (CBM)

The neural networks inside the CBM consist of a large number of *cells* assembled into interconnected *modules*. The cells are the basic processing units of the cellular automata based neural networks. Modules are 3-D lattices of

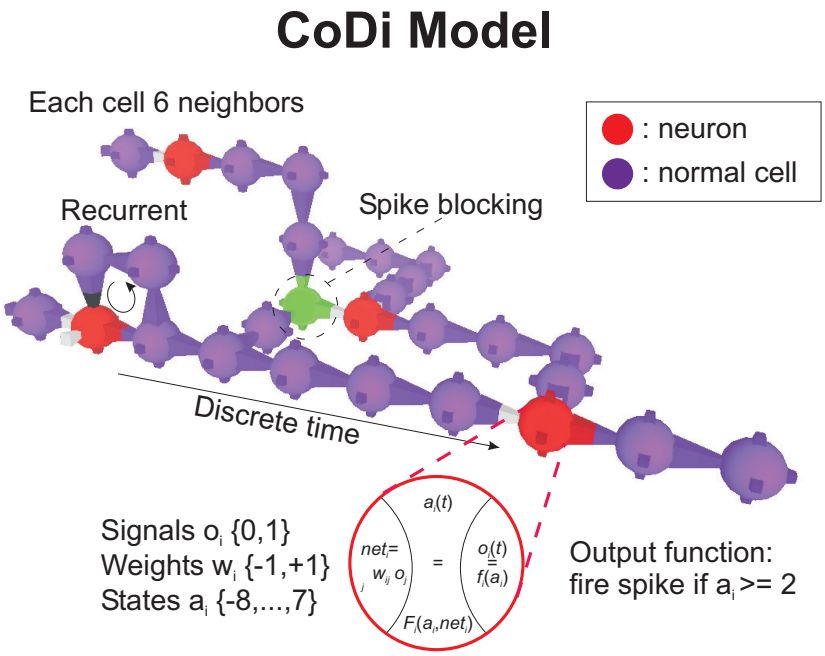

Figure 2. The CBM's neural model. A figure by Rumelhart[4] adapted to the CBM.

$24 \times 24 \times 24$ cells. There are two types of cells: *normal* cells and *neuron* cells. Neuron cells evaluate a non-linear function. They can only occur on fixed positions in the 3-D lattice. The inputs and outputs of neuron cells are so-called 'spiketrains', sequences of one-bit signals. The other, normal, cells transport the spiketrains between the neuron cells. They pass signals from their inputs to their outputs, except when two or more signals arrive at the same time: then they block ('*spikeblocking*') and no signals are transmitted.

In figure 2 the neural model of the CBM is presented. No analog values appear in the model, it is entirely digital. The cells communicate with spiketrains (sequences of 0 or 1, a 1 is called a spike). The inputs of the neurons can be excitatory $(+1)$ or inhibitory (-1). Each neuron has an internal accumulator (discrete state) and fires (emits a spike) when the (weighted) inputs push its value above a threshold.
The connectivity between the cells is very limited. The cells are organized in a 3-D lattice with a von Neumann Neighborhood, so each cell has 6 neighbors.

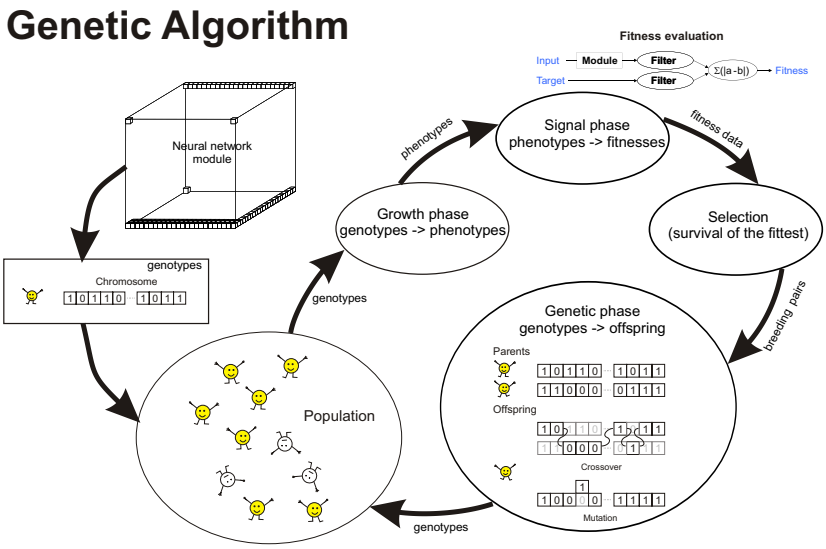

Figure 3. Training of the CBM.

An interpretation of the binary spiketrains is given by the 'Spike Interval Information Coding' (SIIC) representation. By convolving the spiketrains with a convolution filter, they are converted in real time into an analog time-varying signal. Time is very important in the model of the CBM. Signals need time to propagate through the network and it is the time between the signals which encodes the information.

The training of the CBM is supervised and performed off-line. We provide the machine with input spiketrains and want it to produce the given target (output) spiketrains. The CBM learns on a module base, using a genetic algorithm. Since the CBM learns off-line, there are two modes in which it can operate: *evolution mode* and *run mode*. In evolution mode, the machine learns to solve a given problem; in run mode it performs what it has learned.

The training proceeds as follows (fig. 3): initially, a population of *genotypes* of modules is composed. These genotypes are pre-existing encodings of modules. Through a growth phase, these genotypes are transformed into

Table 1. Summary of CBM technical specification

Cellular automata update rate	152 billion cells/s
Max. number of automata cells	893,583,360
Max. number of neurons	74,465,280
Max. number of modules	64,640 ($\approx 2^{16}$)
Max. number of neurons per module	1,152
Max. information flow rate, neuronal level	13.5 Gigabyte/s
Max. information flow rate, inter modular level	74 Megabyte/s
Number of FPGAs (Xilinx 6264)	72
Genotype/phenotype-memory	1.18 Gigabyte
Chromosome length	91,008 bit
Power dissipation	1.5 Kilowatt (5 V, 300 A)

phenotypes, the actual configurations of modules. This growth phase exists to limit the search space and incite better connected neural structures. Then, the training data (input spiketrains) are processed in the Cellular Automata Machine (CAM) with the phenotype configuration and the output is compared with the target. The fitness is evaluated for each candidate and a sequential algorithm, running on an assisting PC, points out a subset of the best modules for further reproduction. In each generation the best modules are mated (crossover) and mutated to produce a set of offspring modules to become the next generation. Everything except the selection of the best modules is executed in hardware on the CBM. This has some major repercussions. The genetic algorithm is kept simple (yet fast). But the exploration of the astronomical search space turns out to be far from sufficiently efficient.

The hardware core of the CBM is the Cellular Automata Machine (CAM) which is able to process one module at a time. All cells inside a module are updated in parallel. In 'Run Mode', the CAM is time-shared between multiple modules. The technical specifications of the CBM are summarized in table 1.

3 Preliminary results

Timing in the CBM is essential; the input spiketrains need some time (latency) to flow through the module and reach the outputs. If we want to compare the output with a target, we have to take this latency into account. This fact has been neglected in previous experiments[5], which has been seriously hindering

Figure 4. An excerpt of the input spiketrain and the desired result (without latency).

the evolvability. This latency is not known in advance, since it depends on the actual topology of the module (the number of cells involved in the longest path). In other words, it is hard to choose appropriate training examples.

Some modifications of the model will be examined, to strengthen the model. Small modifications can relatively easily be realized because the CBM is built up with reconfigurable hardware (FPGAs). Currently we are adopting the supervising software to implement *multiple fitness*. It is an extension of the genetic algorithm to allow the training of network-modules with multiple examples instead of one large concatenation of examples with a fixed order. We want the CBM to generalize from this (arbitrarily chosen) sequence, and not to learn it by heart.

Currently, a few simple networks which can be built manually can also be evolved automatically (with the genetic algorithm). This works for functions as XOR- and NAND-gates, ON/OFF switches and frequency halvers. These are obviously not the ultimate applications; traditional techniques are far more efficient for these tasks. But they deliver some insights in the machine and the learning rule. The results of the manual approach are compared with the automated training technique. This allows us to postulate statements about speed, data representation, accuracy, the complex dynamic behavior of the cellular automata, the number of cells versus the complexity of their signal processing power, etc.

As a simple example the following experiment describes the evolution of a module that fires a spike every time the input changes (fig. 4). We applied a spiketrain of length 672 bits to an input on the module and defined a target with the desired result and a latency of 6 clock ticks. After 21 generations there was a perfect individual in a population of 100 (fig. 5). The solution for this particular problem was found very quickly because it is very easily grown with the growth algorithm.

It took about 4 seconds to prepare all data, 5 seconds to transfer the data to the CBM and 14 seconds to complete the whole genetic evolution.

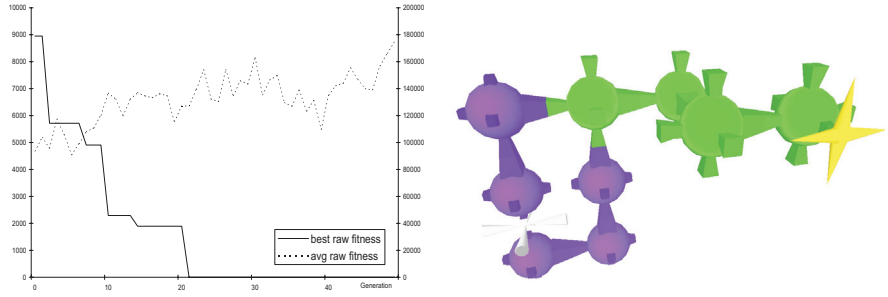

Figure 5. (*Left*) The fitness of the best individual and the average fitness of the population versus the generation number. (*Right*) A visualization of the result. Only the actually used cells are shown (only 9 of the available 13.824). Notice that no neurons are used. The functionality is entirely based on the spikeblocking of the dendrites.

4 Future research

Gradually more complex problems will be tackled; we will move away from machine level into higher abstraction levels, with less handcrafting and more refined training methods. It is not clear yet whether the paradigm of the CBM is directly useful or intermediate models are required. Is it possible to directly program the CBM or are, as in the traditional architectures, stepping stones such as 'assemblers' and 'programming languages' necessary? A possibility is executing the traditional neural networks on the CBM, but then a solution for mapping the analog nature of these networks onto the CBM must be found.
We will work towards two generic problems: a pattern recognition problem and a complex optimization problem (multiple local minima). These are problems in which neural networks typically perform better than traditional techniques, which demand high computational power to deal with such problems. At present there are still many questions about representation, initial and final state, termination of computation. Clearly, many issues are left open for future research.

5 Conclusion

We have described the CBM, its implemented digital neural network model and the preliminary results of experiments, performed on a physical CBM. It

imposes some serious constraints but once a more profound understanding of this peculiar digital model is acquired, its true power will be revealed.

References

1. M. Korkin, G. Fehr, and G. Jeffery. Evolving hardware on a large scale. In *Proceedings. The Second NASA/DoD Workshop on Evolvable Hardware*, pages 173–81. IEEE Comput. Soc., July 2000.
2. D. Talia. Cellular processing tools for high-performance simulation. *IEEE Computer*, 33(9):44–51, September 2000.
3. J.N.H. Heemskerk. *Neurocomputers for Brain-Style Processing. Design, Implementation and Application.* PhD thesis, Unit of Experimental and Theoretical Psychology Leiden University, 1995.
4. D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations. MIT Press, Cambridge, Massachusetts, Londen, England, 1986.
5. H. de Garis, A. Buller, L. de Penning, T. Chodakowski, M. Korkin, G. Fehr, and D. Decesare. Initial evolvability experiments on the cambrain machines (CBMs). In *Proceedings of the 2001 Congress on Evolutionary Computation.*, volume 1, pages 27–30, May 2001.