ATR’s Artificial Brain Project: CAM-Brain Machine (CBM) and Robot Kitten (Robokoneko) Issues

Michael KORKIN (1), Hugo de GARIS (2), N. Eiji NAWA (2), William Dee RIEKEN (2)
(1) Genobyte Inc., 1503 Spruce St., Suite 3, Boulder, CO 80302, USA
http://www.genobyte.com, korkin@genobyte.com
(2) Brain Builder Group, Evolutionary Systems Dept., ATR Research Labs,
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan
http://www.hip.atr.co.jp/~{degaris, xnawa, wrieken}
{degaris, xnawa, wrieken}@hip.atr.co.jp

Abstract—This paper presents some ongoing issues concerning ATR’s Artificial Brain (CAM-Brain) Project. The CAM-Brain Project evolves 3D cellular automata (CA) based neural networks directly in FPGA electronics at electronic speeds in special hardware called a CAM-Brain Machine (CBM). The CBM updates the CA cells at a rate of 150 Billion a second, and can perform a full run of a genetic algorithm (GA) in about 1 second. 32K of these evolved circuits (modules) are then assembled (in a large RAM space updated in real-time by the CBM) into humanly defined architectures to make an artificial brain to control a robot kitten (“Robokoneko”). The paper presents and discusses the latest design decisions for the CBM and the kitten robot, and maps out future plans aimed at having an artificial brain-controlled real robot kitten playing in the ATR labs by early 2001. A world wide, many membered, internet-videophone and electronic pen based, brain-architectural design and brainstorming group will be essential for distributing the design and evolution of the 32K modules. Designing and building an artificial brain within the next three years will be a major conceptual and managerial challenge.

1. INTRODUCTION

This paper discusses ongoing issues concerning ATR’s Artificial Brain (CAM-Brain) Project [1], specifically, the latest design decisions concerning the CAM-Brain Machine (CBM) [3] and the robot kitten (“Robokoneko”). We begin with a general introduction to the CAM-Brain Project as a whole, then describe the CBM design parameters in reasonable detail. The CAM-Brain Machine (CAM stands for Cellular Automata Machine) is a research tool for the simulation of artificial brains. An original set of ideas for the CAM-Brain project was developed by Dr. Hugo de Garis at the Evolutionary Systems Department of ATR HIP (Kyoto, Japan), and is currently being implemented as a dedicated research tool by Genobyte, Inc. (Boulder, Colorado).

An artificial brain, supported by the CBM, consists of up to 32,768 neural modules, each module populated with up to 1,152 neurons, a total of 37.7 million neurons. Within each neural module, neurons are densely interconnected with branching dendritic and axonic trees in a three-dimensional space, forming an arbitrarily complex interconnection topology. A neural module can receive efferent axons from up to 92 other modules of the brain, with each axon being capable of multiple branching in three dimensions, forming hundreds of connections with dendritic branches inside the module. Each module sends afferent axon branches to up to 32,768 other modules.

A critical part of the CBM approach is that neural modules are not “manually designed” or “engineered” to perform a specific brain function, but rather evolved directly in hardware, using genetic algorithms.

Genetic algorithms operate on a population of chromosomes, which represent neural networks of different topologies and functionalities. Better performers for a particular function are selected and further reproduced using chromosome recombination and mutation. After hundreds of generations, this approach produces very complex neural networks with a desired functionality. The evolutionary approach can create a complex functionality without any a priori knowledge about how to achieve it, as long as the desired input/output function is known.

2. CoDi Neural Model

The CBM implements a so called "CoDi" (i.e. Collect and Distribute) [2] neural model. It is a simplified cellular automata based neural network model developed at ATR HIP (Kyoto, Japan) in the summer of 1996 with two goals in mind. One was to make neural network functioning much simpler and more compact compared to the original ATR HIP model, to achieve considerably faster evolution runs on the CAM-8 (Cellular Automata Machine), a dedicated hardware tool developed at Massachusetts Institute of Technology in 1989.

In order to evolve one neural module, a population of 30-100 modules is run through a genetic algorithm for 200-600 generations, resulting in up to 60,000 different module evaluations. Each module evaluation consists of - firstly, growing a new set of axonic and dendritic trees, guided by the module’s chromosome. These trees interconnect several hundred neurons in the 3D cellular automata space of 13,824 cells (24*24*24). Evaluation is continued by sending spiketrains to the module through its efferent axons (external connections) to evaluate its performance (fitness) by looking at the outgoing spiketrains. This typically requires up to 1000 update cycles for all the cells in the module.

On the MIT CAM-8 machine, it takes up to 69 minutes to go through 829 billion cell updates needed to evolve a single neural module, as described above. A simple "insect-like" artificial brain has hundreds of thousands of neurons.
arranged into ten thousand modules. It would take 500
days (running 24 hours a day) to finish the computations.

Another limitation was apparent in the full brain simulation mode, involving thousands of modules interconnected
 together. For a 10,000-module brain, the CAM-8 is capable
 of updating every module at the rate of one update cycle
 1.4 times a second. However, for real time control of a
 robotic device, an update rate of 50-100 cycles per mod-
 ule, 10-20 times a second is needed. So, the second goal
 was to have a model which would be portable into elec-
 tronic hardware to eventually design a machine capable of
 accelerating both brain evolution and brain simulation by a
 factor of 500 compared to CAM-8.

The CoDi model operates as a 3D cellular automata
 (CA). 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 to function as a neu-
 ron, an axon, or a dendrite. Neurons are configurable on
 a coarser grid, namely one per block of 2*2*3 CA cells.
 Cells are interconnected with bidirectional 1-bit buses and
 assembled into 3D modules of 13,824 cells (24*24*24).

Modules are further interconnected with 92 1-bit con-
 nections to function together as an artificial brain. Each
 module can receive signals from up to 92 other modules
 and send its output signals to up to 32,768 modules. These
 intermodular connections are virtual and implemented as
 a cross-reference list in a module interconnection memory
 (see below).

In a neuron cell, five (of its six) connections are dendritic
 inputs, and one is an axonic output. A 4-bit 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 neu-
 ron's chromosome) and either adds to or subtracts from
 the accumulator. The neuron cell's output can be oriented in 6
 different ways in the 3D space. A dendrite cell also has five
 inputs and one output, to collect signals from other cells.
 The incoming signals are passed to the output with an 5-bit
 XOR function. An axon cell is the opposite of a dendrite.
 It has 1 input and 5 outputs, and distributes signals to its
 neighbors. The "Collect and Distribute" mechanism of this
 neural model is reflected in its name "CoDi". Blank cells
 perform no function in an evolved neural network. They
 are used to grow new sets of dendritic and axonic trees
during the evolution mode.

Before the growth begins, the module space consists of
blank cells. Each cell is seeded with a 6-bit chromosome.
The chromosome will guide the local direction of the den-
 dritic and axonic tree growth. Six bits serve as a mask to
 encode different growth instructions, such as grow straight,
turn left, split into three branches, block growth, T-split
 up and down etc. Before the growth phase starts, some
 cells are seeded as neurons at random locations. As the
 growth starts, each neuron continuously sends growth sig-
 nals to the surrounding blank cells, alternating between
 "grow dendrite" (sent in the direction of future dendritic
 inputs) and "grow axon" (sent towards the future axonic
 output). A blank cell which receives a growth signal be-
 comes a dendrite cell, or an axon cell, and further propa-
 gates the growth signal, being continuously sent by the
 root neuron, to other blank cells. The direction of the
 propagation is guided by the 6-bit growth instruction, de-
 scribed above. This mechanism grows a complex 3D sys-
 tem 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. The CBM

This section briefly describes the hardware implementa-
tion of the above CoDi-1Bit model, allowing CoDi neural
 net modules to be grown in hardware.

The CAM-Brain Machine (CBM) was especially de-
signed to support the growth and signaling of neural net-
 works built by the CoDi model. The CBM should fulfill the
 needs for high speeds, when simulating large-scale binary
 neural networks, a necessary condition when one is con-
 cerned with performing real-time control. The hardware
 core is implemented in XC6264 FPGA chips, in which the
 neural networks will actually grow. A host machine will
 provide the necessary interface to interact with the hard-
 ware core. It is planned that the CBM will be used to grow
 32,000 neural networks modules, each with approximately
 1000 cells. The modules will be organized in architectures
 defined in advance, so several neural network modules will
 be interconnected to form a functional unity. For a com-
 plete description of the CBM, refer to [3].

4. Choosing a Representation for the CoDi-1Bit
 Signaling

The constraints imposed by state-of-the-art pro-
 grammable (evolvable) FPGAs in 1998 were such that
 the CA based model (the CoDi model) had to be very sim-
 ple in order to be implementable within those constraints.
 Consequently, the signaling states in the model were made
 to contain only 1 bit of information (as happens in na-
ture's "binary" spike trains). The problem then arose as to
 interpretation. How were we to assign meaning to the
 binary pulse streams (i.e. the clocked sequences of 0s and
 1s which are a neural net module's inputs and outputs?
 Ultimately we chose a representation which convolves
 the binary pulse string with the convolution function shown
 in Fig.2. We call this representation "SIIC" (Spike Inter-
 val Information Coding) which was inspired by [5]. This
 representation delivers a real valued output at each clock
 tick, thus converting a binary pulse string into an analog
 time dependent signal. Our team has already published
 several papers on the results of this convolution represen-
tation work [4]. Figs. 3, 4, 5 and 6 show some results
 of CoDi modules which were evolved to output oscillatory
 signals (using the convolutionary interpretation). Fig. 7
 shows the evolved output for a random target analog sig-
 nal. We thought the results were good enough to settle on
 this representation. The CBM will implement this repre-
sentation in the FPGAs when measuring fitness values at electronic speeds.

Fig. 1. Reproduction of the Robot kitten (robokoneko, in Japanese)

Fig. 2. Decoding filter for the spike trains.

Fig. 3. Single period of a sinusoidal wave generated by the CoDi model and SIIC method. The lower part of the figure show the actual spikes that generated the waveforms.

5. THE ROBOT KITTEN ("ROBOKONEKO") AND RELATED ISSUES

An artificial brain with nothing to control is pointless, so we chose a controllable object that we thought would attract a lot of media attention, i.e. a cute life-size robot kitten that we call "Robokoneko" (which is Japanese for "robo-child-cat") ¹. We did this partly for political and strategic reasons. Brain building is still very much in the "proof of concept" phase, so we wanted to show the world something (that is controlled by an artificial brain) that would not require a PhD to understand what it is doing. If the kitten robot can perform lots of interesting actions, this will be obvious to anyone simply by observation. The more media attention the kitten robot gets, the more likely our brain building work will be funded beyond 2000 (the end of our current research project).

Fig. 1 shows the mechanical design our team has chosen for the kitten robot. Its total length is about 25 cm, hence roughly life size. Its torso has two components, joined with 2 degrees of freedom (DoF) articulation. The back legs

Fig. 4. Two periods of a sinusoidal wave generated by the CoDi model and SIIC method. The lower part of the figure show the actual spikes that generated the waveforms.

Fig. 5. Three periods of a sinusoidal wave generated by the CoDi model and SIIC method. The lower part of the figure show the actual spikes that generated the waveforms.

Fig. 6. Four periods of a sinusoidal wave generated by the CoDi model and SIIC method. The lower part of the figure show the actual spikes that generated the waveforms.

¹For up-to-date data and images on the robot kitten (and the CBM, etc) see the web sites http://www.genobyte.com and http://www.hippair.co.jp/~degaris
have 1 DoF at the ankle and the knee, and 2 DoF at the hip. All 4 feet are spring loaded between the heel and toe pad. The front legs have 1 DoF at the knee, and 2 DoF at the hip. With one mechanical motor per DoF, that makes 14 motors for the legs. 2 motors are required for the connection between the back and front torso, 3 for the neck, 1 to open and close the mouth, 2 for the tail, 1 for camera zooming, giving a total of 23 motors.

In order to evolve modules which can control the motions of the robot kitten, we thought it would be a good idea to feed back the state of each motor (i.e. a spiketrain generated from the pulse width modulation PWM output value of the motor) into the controlling module. Since each module can have up to 92 inputs (actually, 32 inputs repeated 3 times and distributed over three of the input surfaces of the module (minus 4 inputs positions reserved for the 4 output slots) feeding in these 23 motor state values will not be difficult. We are thinking we may install accelerometers and/or gyroscopes which may add another 6 or more inputs to each motion control module. It can thus be seen that the mechanical design of the kitten robot has implications on the design of the CBM modules. There need to be sufficient numbers of inputs for example.

The motion control modules will not be evolved directly using the mechanical robot kitten. This would be hopelessly too slow. Mechanical fitness measurement is impractical for our purposes. Instead we will soon be simulating the kitten’s motions using an elaborate commercial simulation software package called "Working Model - 3D". This software will allow input from an evolving module to control the simulated motors of the simulated kitten. But, does not this approach rather destroy the whole philosophy of the CAM-Brain Machine and the CAM-Brain Project? It is a compromise, certainly, but in practice, the proportion of modules concerned with motion control will be very small compared to the total. Potentially, we have 32K modules to play with. Probably most of them will be concerned with pattern recognition, vision, audition, etc.

6. Future Plans and Challenges

Immediate plans are to use the latest specifications of the CBM to evolve a sample of single modules to show off the CBM’s evolvability (using software simulation until the CBM is delivered). We will use the fitness definition type (i.e. spiketrain comparisons) that will be implemented in the CBM. Once we get a feel for what is evolvable (and we already have quite a lot of experience in evolving CoDi modules in simulation) we will be in a stronger position to start designing and evolving multi module systems. We need to specify a set of behaviors for the kitten robot and then evolve their motion control modules (in simulation). We need to specify what pattern recognition capacities we want. (We have the luxury of 32K modules, so we can afford to be ambitious, provided that the multimodule systems work as well as we hope they will). The first CBM should be delivered to ATR by the end of 1998 (delayed by a year due to a delay by Xilinx in supplying the XC6264 chips).

Section 4 showed how we convert a spike train into an analog signal. We may need to do the reverse, e.g. when a sensor sends an analog signal output voltage (potentiometer output) to an A/D (analog to digital) converter on the kitten, to the kitten’s antenna, and is received by the CBM’s antenna, which then goes through a simple converter which generates a spike train needed for the CBM modules. (CBM modules input and output spike trains). One idea is to evolve a module which takes an 8 bit input stream (a series of byte signals resulting from the A/D converter) and delivers the corresponding pulse train (i.e. if we convoluted it as in section 4, we would end up with the original analog signal). This may be difficult to evolve.

REFERENCES