EVALVABILITY LIMITS
A Case Study Concerning the Modular Evolvable Capacities (MECs) of a New Neural Net Model for a Second Generation Brain Building Machine BM2

Hugo de GARIS

Brain Builder Group, Computer Science Dept, Utah State University, Logan, UT 84322
cell phone +1 435 512 1826, degaris@cs.usu.edu, www.cs.usu.edu/~degaris

Abstract

This paper concerns a case study of the limits to which desirable properties can be evolved in a particular kind of neural network circuit (or module, as it is called here). It has been the decade long dream of the author to evolve neural network modules in their 10,000s at electronic speeds in special evolvable hardware, and then to assemble them into a gigabyte of memory to build artificial brains. But such an dream is only realizable if the (modular) evolvable capacities (MECs) of such modules (i.e. a qualitative and quantitative measure of the quality of the evolution) are sufficiently high to make the effort worthwhile. This paper shows how the evolvable capacities of a module using a new neural network model (called DePo) was stretched to its limits. This paper makes the claim that evolutionary engineering is all about “pushing up MECs”.

1 Introduction

For nearly a decade, the author has been dreaming of building artificial brains by evolving neural net circuits of some 1000 artificial neurons each in special hardware at hardware speeds. In the first attempt to implement these ideas (1996-2001) a cellular automata based neural net model, called “CoDi” [1] was used which was simple enough to be implementable, despite the severe constraints imposed by the programmable (evolvable) state of the art hardware of 1996 (the Xilinx 6200 family of FPGAs (field programmable gate arrays)). The first generation brain building machine, called “CBM” (CAM-Braṇi Machine) (CAM = cellular automata machine) was implemented and sold to Japan, Europe, and the US (4 machines). Unfortunately the author’s previous research lab went bankrupt and did not complete payment of the CBM supplied to his lab. At the time of writing (November 2001) it is not clear what the fate of the CBM will be, as developments concerning this machine are still unfolding. It is a research machine and needs further work. However, its chips date from 1996 and are starting to age. Independently of whether the CBM research and development work will continue or not, it is time to start serious thinking about the creation of a second generation machine, which we call “BM2” (Brainbuilding Machine, 2nd generation). The author is now a professor of computer science in the US and has been hired to establish a “brain building center” at his university. Starting in 2002, he will teach the world’s first PhD course on the principles of brain building, so it is appropriate that such “second generation” work should begin.

The basic idea behind brain building is to evolve neural net circuits at electronic speeds, i.e. at a rate of about one evolved neural net in a second or so, and then to assemble tens of thousands of them into artificial brains. The elite evolved neural net can then be downloaded into a gigabyte of RAM. Tens of thousands of such individually evolved neural net “modules” can be downloaded. Human “brain architects” (BAs) then interconnect these modules to make humanly specified artificial brain architectures. The same machine (the CBM) which was used to evolve the individual modules is then employed to perform the signaling of the assembled RAM based artificial brain, in real time, so that the brain can be used, for example to control the behaviors of a robot and to give it thousands of different pattern detector circuits, decision making circuits, and hundreds of motion control circuits. This is the dream. Modern electronics makes it possible, and the probability that Moore’s
law will continue for another 20 years or more, makes the prospect of brain building almost certain.

However, if one is to perform “evolutionary engineering” (i.e. the construction of complex systems using an evolutionary (Darwinian) approach) then it is critical that the systems that evolve are capable of performing the tasks desired of them. The researchers in the field of evolutionary engineering have become increasingly conscious over the past few years of the importance of the concept of “evolvability”, i.e. the ability of the systems being evolved to actually perform the task required of them. Since there is no theory of evolvability, an empirical “hit or miss” approach is used. As any evolutionary engineer (EE”) knows, there is no certainty that a desired functionality will actually evolve. Even if the desired function does evolve, it may not evolve well, i.e. with high quality. The author has coined the term MEC (Modular Evolvable Capacity) [3] to express a quantitative measure of the quality of the evolution, within a given context (for example, for how long can an emitted analog signal from an evolved neural net, follow some time varying target signal before it inevitably diverges). The author has become increasingly conscious that neural net models need to be chosen which prove to have high evolvabilities, so that they may perform well when implemented in programmable/evolvable electronics. This paper presents a new neural net model with superior evolvabilities compared to the neural net model implemented in the previous generation brain building machine, CBM.

Despite the fact that this new model has superior evolvabilities, it will have its limits, i.e. its evolvable capacity (EC). Since we are talking about the EC of a module (i.e. a neural network of a given size), we talk about the module’s MEC (modular evolvable capacity [4]). If it is not clear that any module will have its MEC, consider the following argument. Imagine a module capable of being evolved such that its output is able to follow a random time varying target curve for an infinite amount of time. It takes an infinite number of bits of information to specify the shape of the infinitely long target curve, yet the module is evolved with a finite number of bits. Experience shows [4] that by increasing the number of bits used in the evolution (i.e. the number of bits used to specify the model) then usually, the MEC of the module increases. It appears as though there is some form of (new?) conservation law manifesting itself here. You only get out in MEC what you put in in the number of bits in the model. Hence any module will have a finite MEC. Once this MEC is reached, the module fails to evolve. This paper introduces a new neural net model, called DePo, and shows that it has a higher MEC, i.e. it follows a target output curve more accurately and for longer than an earlier model, called CoDi, but it too fails in its turn. The DePo model was evolved with increasingly longer target curves until its MEC was reached, as will be shown in the figures later in this paper.

The contents of the remainder of this paper are as follows. Section 2 describes briefly the previous neural net model (called “CoDi” (Collect and Distribute)). This model will serve as a basis for comparison with the new “DePo” (Delayed Pointer) neural net model introduced in this paper. Section 3 presents some lessons learned from the results of CoDi model evolution, which motivated the choice of some of the characteristics of the DePo model. Section 4 presents details of the DePo model. Section 5 presents some early results of DePo module evolution. Section 6 contains some discussion and plans for future research, especially plans for building and financing the second generation brain building machine BM2. Also discussed are the implications of finite MECs of any neural net model used, and how this fact will impact on any brain building project that uses the evolutionary engineering approach, as this one does and will.

2 The CoDi (Collect and Distribute) Neural Net Model

State of the art programmable (evolvable) hardware in the year 1996 was Xilinx’s XC6200 family of chips. 1996 was the year that design on the 1st generation brain building machine CBM was begun. The constraints imposed by the hardware implied that the simplicity of the neural net model that the CBM implemented had to be fairly simple. For example, the number of bits used in the neural signal values was restricted to 1. (Every extra bit would have implied the use of an extra wire in the silicon). The neural model finally implemented in the CBM was called CoDi [1] which will be described briefly in this section. There have been many papers published on this model, see for example [1-3]. The CoDi neural net model is 3D cellular automata based. Axons and dendrites grow in a 24*24*24 cube of 3D CA cells under genetic control. Each cell is given initially 6 growth enable bits, one for each face of a 3D CA cell (a little cube). Growth is synchronous (i.e. all cells will change their states, if enabled, at the same clock tick). If, at tick T, a particular cell is a dendrite cell, then a neighboring blank cell will become a dendrite cell at tick T+1,
if the enabling growth bit on the contiguous face of the parent cell is set to 1. A newly grown cell sets a “parent pointer” in the direction of its immediate parent cell. Hence all growing axons and dendrites have parent pointers (PPs) which point back to their originating neurons. These parent pointers are used in the signaling phase to instruct the binary signals in which direction to move (for example, a binary signal will follow the parent pointers if it is in a dendrite). A similar story holds for axon signals. When a signal arrives at a junction (a synapse) between an axon and a dendrite, it crosses over from the axon to the dendrite. When a signal arrives at a receiving neuron it is multiplied by either +1 or -1. These signal weightings are under genetic control. Each neuron has a 4 bit counter. The sum of the weighted binary signals is added to the value already present in the counter from the previous clock tick. If the new counter value becomes 2 or more, the neuron fires, i.e. places a 1 on all its axons. (At the 1st clock tick in the growth phase, axons grow out of a neuron if the growth enable bits are 1s, otherwise dendrites, so a neuron can have up to 6 axons or 6 dendrites and all combinations in between). If a neuron fires, its counter is set to 0. If the counter value goes below -7, it simply resets to 0. A 24*24*24 cube of 3D CA cells is called a “module”. A module has up to nearly 200 input points and only 4 (fixed position) output points. (Actually, usually only one output point is used.) The binary output signal string can be convoluted with an integer based convolution function [3] to convert it into an analog signal. The modules are then evolved so that the binary or analog output signals match as closely as possible some target output signal.

A few actual experiments were performed on the author’s CBM (i.e. the actual hardware) before it was switched off (December 2000) by his hardware colleague (the builder of the 4 CBMs in the world) due to incomplete payment when the author’s previous lab was going bankrupt. The most significant experiment was probably the evolution of a module which gave a strong output signal when a “line” of stimulation moved up an input face of the 24*24*24 cube, and a low signal when the line moved down [4]. Hubel and Wiesel got a Nobel Prize in the 1970s for similar work when they discovered that a comparable process occurs in the kitten’s brain. It was incredibly frustrating, after so many years of anticipated use of the CBM, that it was only working for a few months before it was switched off. (The author’s hardware colleague had remote access via the internet to constantly update the CBM’s firmware). Other experiments, such as measuring a module’s MEC (Modular Evolutionary Capacity, e.g. for how long could an evolved output signal follow closely a time varying target signal) were performed. Fig. 1 shows some of these early results. The impression is that the quality of the evolution (the evolvability) is encouraging, but it would be nicer if the accuracy were better. Hence the current attempt to increase the evolvabilities of neural nets with more evolvable models.

3 Remarks on the CoDi Neural Net Model

The CoDi model has some obvious restrictions. For example, a functional neuron can receive maximum 5 input bits per clock tick. This is indeed a rather severe restriction, and probably influences strongly the evolvability of a CoDi module. There is also an inevitable delay between the moment of input of a signal and its output, due to the travelling time (one CA cell little cube per clock tick) of the signal through the large cube. If one were to build an artificial brain based on CoDi modules, there would be a risk that when the outputs of one module are connected to the inputs of other modules, the total delay or response time, would risk being large. The CoDi model also has many 3D CA cells that are not used, yet they are processed at every clock tick to see if their state has changed. This is inefficient. It would be better to have a model in which all components of the model are used.

These restrictions and other considerations motivated the author to invent a new neural net model that would be more complex than the CoDi model (taking advantage of Moore’s law, which has had 3 doublings over the past 5 years). The 5 bit input per neuron per clock tick restriction seemed particularly heavy, so one immediate suggestion for a new model was to have many inputs come into a neuron at each tick, e.g. hundreds. (The author has always been puzzled why nature has many thousands of synapses on its biological neurons, or why a protein has no many atoms when only a small portion of its surface is the “active site”). The suspicion is that these large number of components allows for greater evolvability. If a signal coming into a neuron changes, the overall effect on the neuron may be small. In evolutionary computation terms, the suspicion is that with many inputs (hundreds) the fitness landscape of the evolving neural nets will be very smooth and hopefully more evolvable. It would also be desirable to have multibit signalling, so that the value of the output signal could be simply “read off”
by interpreting the output multibit signal as a binary representation of an integer (or float). Another idea which motivated the design of the DePo model was the desire for higher MECs. By evolving long delay signaling times between neurons, it was hoped that longer MECs would become possible. A combination of the above considerations motivated the DePo model which is now described in more detail.

4 The DePo (Delayed Pointer) Neural Net Model

4.1 Basic Ideas

As the name of the new model suggests, its two essential features are its use of pointers (to the other neurons that an individual neuron connects to) and delays (the number of clock ticks it takes for a neural signal to be transmitted from its emitting neuron to its receiving neuron). Both pointers and delays are under genetic control. As hinted to above, a neuron can emit and receive hundreds of signals. These signals are real valued, signed (+ or -), of the form +/- 0.yyyyy where the yyyy is a binary fraction. When large numbers of signals arrive at a neuron, due to the more or less equal number of both +ve and -ve signal values, they will tend to cancel each other out (taking advantage of the law of large numbers). If the values of the weights are not too large, it is possible that the sums of these incoming signal values multiplied by the weights on each connection (in math terms, the dot product of the weight vector of a neuron and its incoming signal vector at a given clock tick T) do not blow up exponentially, so that a transmission function (e.g., a sigmoid function) is not needed to be applied to this dot product. (Actually when this hypothesis was tested, it was found to be false. It always blew up, so a transmission function was eventually employed.) Thus this dot product value “x” (modified by the transmission function y(x) = (2/(1 + exp(-x)) -1 was passed to all the neurons that connect with it.

Each neuron is given a unique integer identifier (e.g., if there are N neurons in the network, then label them from 0 to N-1). Under genetic control, generate for each neuron a TO-List, i.e. a list of neuron identifiers of the neurons to which the neuron will send its output signals to. From these TO-Lists it is possible to deduce each neurons FROM-List (TO[i,j] = FROM[j,i]) i.e. a list of neuron identifiers of those neurons from which the neuron receives its input signals. The FROM-List is used by each neuron to calculate its signal weight.

The TO-List is used to emit the dot product neural signal to all those neurons that the neuron connects to. If a connection exists between two neurons “i” and “j”, then an entry is created in a LINK-Delay-List. Each entry points to a 1D array of Dij real numbers which are the signal values as they are transmitted to the receiving neurons. Each signal moves one position in the array per clock tick (actually, emit and absorb pointers are moved backwards (with wraparound along this array). The sizes Dij of these arrays are under genetic control. The Dij values are the delays, i.e. the number of clock ticks it takes for a neural signal emitted by neuron “i” to be absorbed by neuron “j”. These “delay links” or “delay lines” are a kind of constant length queue, and were implemented in software simulation in the following way. At initialization time, the emit pointer for each link was set at the 0 position, and the absorb pointer was set at the Dij-1 position. The order of processing at each clock tick was as follows. Firstly, calculate the dot product for each neuron, i.e. pick up the neural signal values at all the absorb pointer positions and multiply them by the corresponding weights associated with the links. Secondly, shift the pair of pointers (emit and absorb) backwards along the link, which is equivalent to the signals in the array moving forwards along the link). Thirdly, place the newly calculated dot product (modified by the transmission function) at all the positions pointed to by the emit pointers on all the links that the neuron connects to. An external input signal value (unweighted), (if there is one for an individual neuron) is added to this modified dot product. Some neurons (of the N) are specified to be output neurons. The signals they output are sent also to the outside world. These signal values are easy to interpret. Their multibit representation is that of a binary fraction (0,yyyy).

4.2 Genetic Operators

The choice of the genetic operators which were thought to be applicable to the DePo model was pretty much what you would expect. They are listed here and described briefly. The chromosome contained 2 subfields, i) the weights of all the links, ii) the delays of all the links.

The genetic operators actually so far implemented are -

a) Increment a Delay (for a given LINK-Delay, increment the delay, i.e. Dij++). b) Decrement a Delay (Dij–). c) Create a new random Delay (Dij). d) Mutate a Weight (for a given link[i,j], apply a bit flip in a random position along the binary representation of
the weight 0.4444 - the sign can also be independently flipped.

Operators not (yet) implemented could include -
  a) Add a Neuron (which adds the (N+1)th neuron to the network, which in turn generates its own TO-
    List and LINK-Delay-List, with weights of very low values, so as not to perturb the network dynamics too
    much. b) Delete a Neuron (remove the appropriate pointers to/in the TO-List, LINK-Delay-List, etc) c) Add
    a Link (if a link does not already exist between neurons i and j, then create a link between them, add it
    to the LINK-List, generate its delay Di, and its weight (of low value)). d) Delete a Link (if a link already
    exists between neurons i and j, then remove it from the LINK-List, update the corresponding pointers etc).

5 Experimental Results

Fig. 1 shows the target curve (Series 1) and the evolved curve (Series 2) over 150 clock ticks using the
older CoDi model. Fig. 2 shows a similar result using the new DePo model. The figures speak for themselves. The CoDi model was unable to follow the target curve closely as the latter varied in time over the
150 clock ticks. The DePo model on the other hand was able to do this far better.

In the actual implementation, the network was fully connected, (i.e. with N*N links between N neurons,
where typically N was 100 to 300). Adding/deleting of neurons and links was not undertaken in the limited
150 time available, but will be in the future. Fig. 2 shows the results where the maxdelay was 150, N = 100,
there were 8 bits in the signed weights, an exponential transmission function (ranging in output between +1
and -1, y(x) = (2/(1 + exp(-x))) -1) was applied to the weighted and summed input signals. The transmission
function was essential for evolvability. A simple +1 or
-1 saturation value (no transmission function) simply saturated most of the net and gave no handle for evolu-
tion. The fitness measure used was a simple inverse
of the sum of the squares of the differences between the
values of the target and evolved curves over T clock
ticks.

Seeing the relative success of the DePo model for
150 clocks, the author was curious to see what the DePo model’s MEC would be (i.e. for how many clock
ticks could the actual evolved output curve follow reason-
ably accurately the target curve). The next experi-
ment was with 250 clocks, using the same target curve
(whose formula was y(x) = 0.52 + 0.25sin(2Pix/60) -
0.15cos(2Pix/40) + 0.1sin(2Pix/50). This experiment
failed, i.e. the fitness values got stuck at low, unac-
ceptable levels with poor performance. So in the next
experiment, a target curve length of 200 clock ticks
was tried - with the same negative result. Obviously,
this module’s MEC was less than 200. Since the 150
clocks experiment worked reasonably well, a 175 fig-
ure was tried. The MEC of the module must be near
this number. Fig. 3 shows the module is struggling to
evolve an accurate output curve. It is almost failing.

Fig. 1 The evolved output and target curves for the
CoDi model, 150 clock ticks

Fig. 2 The evolved output and target curves for the
DePo model, 150 clock ticks

6 Discussion and Future Research

Despite our pleasure at witnessing the consider-
ably superior evolvability of the new DePo model,
a lot more work needs to be done on it to discover
its strengths and limitations. Once this has been
done, the next main stage in the research can be-

a...
grammable/evolvable hardware to start building the 2nd generation brain building machine BM2. It will be interesting to see to what extent we will be forced to compromise the DePo model to fit it into state of the art programmable chips. (DePo’s high evolvability in Fig. 2 suggests that it will be also very evolvable for other tasks, such as pattern recognition). Very probably the chips to be used will be chosen from the “Virtex” family of Xilinx FPGA chips. The top of the range of these chips, with its 8 MILLION GATES costs up to 10,000 dollars each, so obviously, we will be using the cheaper smaller versions. At the time of writing (Nov 2001), a start up company (in which the author is a scientific consulting director), called SENAPPS (“Sentient Applications”, with Steve Long as CEO [www.senapps.com]) based in Salt Lake City (an hour and a half drive from the author) is busy raising seeder capital. It is Steve Long’s desire to see SENAPPS become the world’s first artificial brain company. Starting in January 2002, the author will be teaching the planet’s first “brain building” course to M.Sc. level, and 6 months later, to PhD level students, so there will be plenty of research workers to build the brain. There is also a very active robotics group of about 30 people at the author’s university with very competent hardware people to help with the construction of the BM2. Once the DePo or a simplified model has proven to be useful enough to be placed in hardware, multimodule simulation experiments will be undertaken, to gain experience in designing artificial brains. The author was hired by the head of the computer science department at Utah State University (USU) with the basic aim in mind of establishing a “BBC” (a Brain Building Center) or “BBI” (Brain Building Institute), the first of its kind in the world. It will consist of research workers, students, professors, and companies, all with the same dream of building artificial brains, an activity that modern day electronic capacities is making increasingly realistic and doable (e.g. with 8 MEGAGATES in a single chip!) But, to end on a note of caution - the above optimism is all very fine, but there are conclusions to be drawn from Fig. 3. Any module will have its MEC. This raises some interesting speculative questions. Is it possible to get better MECs with the same number of bits being input into the evolution of the model? Is there an inevitable price to be paid for higher MECs, namely the higher number of bits put into the model? Are we talking here of a “Conservation of Evolvability”, a new kind of conserved quantity? The author will soon have many students in his brain building class who can play with different neural net models. If empirical experience shows that there seems to be such a (new?) conservation law, then any brain building team in the future will simply have to supply the necessary number of bits to generate the required MECs. The price to be paid for higher MECs will be a proportionately greater cost in terms of the area of silicon needed to implement the model in electronics. Or, is it possible (by working smarter?) to achieve more MEC for less bits? The question remains interesting and open. Perhaps by increasing the number of bits in the weights, and the signals, MECs will increase? Further experimentation needs to be done.

References