The CNN is Universal as the Turing Machine

Leon O. Chua, Tamás Roska, and Péter L. Venetianer

Abstract—It is shown that the game of life algorithm, which is equivalent to a Turing machine, can be realized by CNN; thus the CNN is also universal.

I. INTRODUCTION

It is well known that in the realm of logic computing, Turing machines are universal in the 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].

The cellular neural network (CNN) is a new paradigm connected, geometrically placed, analog, 3-D regular processing array. By introducing the nonlinear and delay-type templates and additional capabilities [3] we have an extremely broad universe of functionalities. Multilayer perceptrons can be realized also by a class of CNN's [4]. It can be shown that all the three types of partial differential equations can be approximated by CNN's. The stored program CNN's [10] provide a new quality: a simple way to solve CNN "analogic" 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 [6] with CNN "analogic" algorithms, a kind of analog software.

In this paper we show that the game of life can be realized by CNN's, which means that any Turing machine can be realized by simple programmable CNN's, thus, we can prove that the programmable CNN is, theoretically, a universal machine in both the analog and the logical field.

II. THE RULES OF THE GAME

Being a no-player game, game of life [7] is not really a game in its traditional meaning. In principle it is "played" on an infinite square 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 by overcrowding: a cell that is alive at time $t$ and has more than 3 living neighbors at time $t$ will be dead by time $t + 1$.

Death by exposure: a cell that is alive at time $t$ and has less than 2 living neighbors at time $t$ will be dead by time $t + 1$.

Survival: a cell that was living at time $t$ will remain alive at $t + 1$ if and only if it had exactly 2 or 3 alive neighbors at time $t$.

These simple rules result in surprisingly complex behavior: a life pattern can fade away or stay alive in many different ways, for example becoming static, oscillating or growing infinitely. Moreover, given an initial state, to decide its destiny is equivalent to the halting problem of Turing machines.

III. THE BASIC STEP TEMPLATE

The fact that the cells of the game of life array can be projected onto those of the CNN and the new state of a cell is determined only by its previous state, and that of its neighbors, suggest that the CNN may be designed to implement the game of life algorithm in real time. In order to do this, let us reformulate the rules as follows: a cell will be alive if and only if at least 3 of the 9 cells in its $3 \times 3$ neighborhood are alive and at most 3 of its 8 neighbors are alive; otherwise, it will be dead. The two parts of the conjunction can be easily transformed into CNN templates. Moreover, the conjunction itself can be put together with one of the templates, resulting in the following 2-layer template, where layer 1 corresponds to the first, layer 2 to the second half of the conjunction, and to the conjunction itself (see Fig. 1 for function $\sigma$ in template $A_{12}$):

$$A_{11} = [1] \quad B_{11} = \begin{bmatrix} 0.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix} \quad I_1 = 1$$

$$A_{12} = [a]$$

$$A_{22} = [1] \quad B_{22} = \begin{bmatrix} -0.6 & -0.6 & -0.6 \\ -0.6 & 0 & -0.6 \\ -0.6 & -0.6 & -0.6 \end{bmatrix} \quad I_2 = -0.8$$

Hence, it seems that the programmable CNN, in the above sense, is universal also in the Turing sense.

Manuscript received February 3, 1993. This work was supported the National Research Fund of Hungary under Research Grant 25789/1, and by the National Science Foundation and the Hungarian Academy of Sciences under Joint Grant INT 90-01336. This paper was recommended by Associate Editor Y. F. Huang.

Reference:


Fig. 1. The function used in the 2-layer template of an analog CNN for executing one step of the game of life.

Fig. 2. The single-step life algorithm. (a) input pattern, (b) output of layer 1, (c) new state (output of layer 2).

IV. VERSIONS OF A MULTISTEP CNN ALGORITHM

4.1. Discrete Time CNN with Thresholding Sigmoid

The algorithm described in the preceding section executes a single step of the game of life with one transient. The next step is to simulate a whole life history with one CNN, i.e., the transient will settle down only if the game has reached a stable state; otherwise, it would run forever. To reach our goal, we have to feed the output back to the input, but only after all cells are in their new stable states. This means that we have to time the feedback somehow. This can most easily be done by using a discrete time cellular neural network (DTCNN) [8] with a sigmoid thresholding, which requires only a single step to produce a new output. The 2-layer solution described above needs to be slightly modified: three layers are now required, one for each part of the conjunction and a third layer to realize the conjunction itself, yielding the output pattern.

\[
A_{31} = \begin{bmatrix} 0.3 & 0.3 & 0.3 \end{bmatrix} \quad A_{13} = \begin{bmatrix} 0.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix} \quad I_1 = 1
\]

\[
A_{32} = \begin{bmatrix} -0.6 & -0.6 & -0.6 \\ -0.6 & -0.6 & -0.6 \end{bmatrix} \quad A_{23} = \begin{bmatrix} 0.3 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.3 \end{bmatrix} \quad I_2 = -0.8 \]

\[
A_{33} = \begin{bmatrix} -0.6 & -0.6 & -0.6 \end{bmatrix} \quad I_3 = -1.5
\]

4.2. Discrete Time CNN with Piecewise-Linear Thresholding

Another solution can be achieved with a DTCNN using piecewise-linear thresholding, just as 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 [3] we can extract the coded information. The following template realizes the multistep game of life algorithm in one layer (see Fig. 3 for the functions of template A):

\[
A = \begin{bmatrix} a & a & a \\ a & b & a \\ a & a & a \end{bmatrix} \quad B = 0 \quad I = 0.
\]

The first step of the algorithm is to code the information about the previous state and the number of living neighbors of the current cell. 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:

\[
\text{base number} + 0.1 \times \text{number of living neighbors}
\]

where the base number is -0.5 for the white (-1) cell, and -0.45 for a black (+1) cell. The function b computes the base number and the function a computes the number of living neighbors.

Consequently, in the second step, the decoding phase will 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.

REFERENCES

Comments on “Frequency-Domain Conditions for the Robust Stability of Linear and Nonlinear Dynamical Systems”

F. G. Boese

Abstract—The title’s paper is commented on and several corrections are given.

I. INTRODUCTION

Given a set \( D \subset C \). Given further a parametric family,
\[
F_2 = \{ f_p(s) : p \in \mathcal{P} \} \tag{1.1}
\]
of functions \( f_p(s) \) of the complex variable \( s \) holomorphic in \( D \) depending continuously on a real or complex parameter vector \( p = (p_1, p_2, \ldots, p_u) \) of finite dimension \( u \in N \). The vector \( p \) ranges over a closed region \( \mathcal{P} \) in an embedding topological Hausdorff space \( P \).

Several situations in system theory give rise to consider the problem of the theory of functions of one complex variable to find criteria by which one can effectively verify whether or not
\[
f_p(s) \neq 0 \quad \text{in} \quad D \quad \text{for} \quad p \in \mathcal{P} \tag{1.2}
\]
This question received much attention in the engineering community since the seminal papers [9], [10] of Kharitonov. The authors of the above paper try to filter from the existing body of literature the fundamentals of those techniques which work in an image plane.

Manuscript received August 5, 1991. This paper was recommended by Associate Editor M. Hasler.

The author resides at D-81373 München 70, Germany.

IEEE Log Number 9208418.

2Up to 1990, Prof. V. L. Kharitonov wrote 10 papers with a total of 35 pages, not counting his doctoral thesis of 240 pages from 1989.