



Optimizing Time-multiplexing Raster Cellular Neural Network simulator using genetic algorithms with RK8(6)

KEYWORDS

single-layer Cellular neural networks, genetic algorithms, RK4(2),RK4(3) and RK8(6)

O. H. Abdelwahed

M. El-Sayed Wahed

O. Mohamed Eldaken

Department of Computer Science
Faculty of Computers and Information, Suez Canal University, EGYPT

ABSTRACT In this research, a versatile algorithm for simulating CNN arrays and time multiplexing is implemented using numerical integration algorithms; RK4(2), RK4(3) and RK8(6) and. The approach, time-multiplexing simulation, plays a pivotal role in the area of simulating hardware models and testing hardware implementations of CNN. Owing to hardware limitations in practical sense, it is not possible to have a one-one mapping between the CNN hardware processors and all the pixels of the image. The simulator is capable of performing CNN simulations for any size of input image, thus a powerful tool for researchers investigating potential applications of CNN. This paper presents proper CNN templates for edge detection in image processing. Training of CNN is done by using genetic algorithm

1 Introduction

In this paper, we focus on edge detection which is much significant in various application as virtual reality, intelligent human-computer interface and TV-conference, security system.[18,19,20]. First step in edge detection is locating face and facial features [18],[19]. Then, the detected edges have to be normalized and recognized by specially designed classifier. In this study, we find the facial features and face region using CNN templates estimated by using genetic algorithms.

Most of the widely applied genetic cellular neural networks fall into two main classes: (1) memoryless cellular neural networks and (2) dynamical cellular neural networks. As in Hopfield networks (HN) and CNN, dynamical neural networks have usually been designed as dynamical systems where the inputs are set to some constant values and each trajectory approaches one of the stable equilibrium points depending upon the initial state. Cellular Neural Network is a large-scale non-linear analog circuits which processes signals in real time[2]. The network behavior of CNN depends on the initial state of the cells activation, namely bias I and on weights values of A and B matrices which are associated with the connections inside the well-defined neighborhood of each cell. There are many approaches in estimation of A,B, I matrices. Here we prefer genetic algorithm. Genetic algorithm is a learning algorithm based on the mechanism of natural selection and genetics, which have proved to be effective in a number of applications. It works with a binary coding of the parameter set, searches from a number of points of the parameter space. It uses only the cost function during the optimization, it need not derivatives of the cost function or other information. [16-22].

2 Cellular Neural Networks

Like cellular automata, the CNN is made of a massive aggregate of regularly spaced circuits clones, called cells, which communicate with each other directly only through nearest neighbors. In Figure 2, each cell is modeled as squares. The adjacent cells can interact directly with each other. Cells not directly connected together may affect each other indirectly because of the propagation effects of the continuous-time dynamics of cellular neural networks. An example of a two-dimensional CNN is shown in Figure 3. Now let us define the neighborhood of C(i,j).

Definition : r-neighbourhood

The r-neighbourhood of a cell C(i,j) , in a cellular neural network is defined by,

$$N_r(i, j) = \{C(k, l) \mid \max\{|k-i|, |l-j|\} \leq r\} \quad (1)$$

$1 \leq k \leq M; 1 \leq l \leq N$

where r is a positive integer number.

Figure 3 shows neighborhoods of the C(i,j) cell (located at the center and shaded) with r=1 and 2, respectively. To show neighbourhood relations more clearly, the center pixel is coloured as black and related pixels in brown in Figure 3 and 4. Cells are multiple input-single output nonlinear processors all described by one, or one among several different, parametric functionals. A cell is characterized by a state variable, that is generally not observable as such outside the cell itself. It contains linear and non-linear circuit elements such as linear resistors, capacitors and non-linear controlled sources (Figures 4 and 5).

Every cell is connected to other cells within a neighborhood of itself. In this scheme, information is only exchanged between neighbouring neurons and this local information characteristic does not prevent the capability of obtaining global processing. The CNN is a dynamical system operating in continuous or discrete time. A general form of the cell dynamical equations may be stated as follows:

$$C \frac{dx_i(t)}{dt} = -x_i(t) + \sum_{u \in N_r(i)} A_{(i-u)(j-v)}(t) y_u + I \quad (2)$$

$$y_i(t) = \frac{1}{2} (|x_i(t)+1| - |x_i(t)-1|)$$

where x,y,u,l denote respectively cell state, output, input, bias and j and k are cell indices. CNN parameter values are assumed to be spaced-invariant and the nonlinear function is chosen as piece-wise linear (Figure 5). Since we use discrete 2-D images, Equation (2) is rewritten as,

$$x_i(n+1) = -x_i(n) + \sum_{u \in N_r(i)} A_{(i-u)(j-v)}(n) y_u + I \quad (3)$$

$$y_i(n) = \frac{1}{2} (|x_i(n)+1| - |x_i(n)-1|)$$

with A, B and I being cloning template matrices that are identically repeated in the neighbourhood of every neuron as,

$$A = \begin{pmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,-1} \\ a_{1,-1} & a_{1,0} & a_{1,-1} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,-1} \\ b_{1,-1} & a_{1,0} & b_{1,1} \end{pmatrix} \cdot I \quad (4)$$

The network behaviour of CNN depends on the initial state of the cells activation, namely bias I and on weights values of A and B matrices which are associated with the connections inside the well-defined neighbourhood of each cell. CNN's are arrays of locally and regularly interconnected neurons, or, cells, whose global functionality are defined by a small number of parameters (A, B, I) that specify the operation of the component cells as well as the connection weights between them. CNN can also be considered as a nonlinear convolution with the template. Cells can be characterized by a functional block diagram that is typical of neural network theory: Figure 4 depicts a two-stage functional block diagram of a cell, composed of a generalized weighted sum (in general nonlinear with memory) integration, output nonlinear function/functional. Data can be fed to the CNN through two different ports: initial conditions of the state and proper input u . Bias values I may be used as a third port.

The network behaviour of CNN depends on the initial state of the cells activation, namely bias I and on weights values of A and B matrices which are associated with the connections inside the well-defined neighbourhood of each cell. CNN's are arrays of locally and regularly interconnected neurons, or, cells, whose global functionality are defined by a small number of parameters (A, B, I) that specify the operation of the component cells as well as the connection weights between them. CNN can also be considered as a nonlinear convolution with the template. Since the introduction of Chua [2], CNN has attracted a lot of attention. Not only from a theoretical point of view these systems have a number of attractive properties, but furthermore there are many well-known applications like image processing, motion detection, pattern recognition, simulation. The reduced number of connections within a local neighbourhood, the principle of cloning template etc., turn out to be advantage of CNN's.

3 Behavioral Simulation

Recall that equation (1) is space invariant, which means that $A(i,j;k,l) = A(i-k,j-l)$ and $B(i,j;k,l) = B(i-k,j-l)$ for all i,j,k,l .

Therefore, the solution of the system of difference equations can be seen as a convolution process between the image and the CNN processors. The basic approach is to imagine a square subimage area centered at (x,y) , with the subimage being the same size of the templates involved in the simulation. The center of this subimage is then moved from pixel to pixel starting, say, at the top left corner and applying the A and B templates at each location (x,y) to solve the differential equation. This procedure is repeated for each time step, for all the pixels. An instance of this image scanning-processing is referred to as an "iteration". The processing stops when it is found that the states of all CNN processors have converged to steady-state values[2] and the outputs of its neighbor cells are saturated, e.g. they have a +1 value.

This whole simulating approach is referred to as raster simulation. A simplified algorithm is presented below for this approach. The part where the integration is involved (i.e. calculation of the next state) is explained in the Numerical Integration Methods section.

In the following two subsections we will discuss genetic algorithms and edge detection by CNN and genetic algorithms.

3.1 Genetic Algorithms

In the estimation of A, B , and I matrices of CNN, we use genetic algorithms.

Genetic algorithm is a learning algorithm based on the mechanism of natural selection and genetics, which have proved to be effective in a number of applications. It works with a binary coding of the parameter set, searches from a number of points of the parameter space. It uses only the cost function during the optimization, it need not derivatives of the cost function or other information. [11,12]. Processes of natural selection cause chromosomes that encode successful structures to reproduce more often than those that do not. In addition to reproduction, mutations may cause the chromosomes of children to be different from those of their biological parents, and crossing over processes create different chromosomes in children by changing the some parts of the parent chromosomes between each other. Like nature, genetic algorithms solve the problem of finding good chromosomes by manipulating in the chromosomes blindly without any knowledge about the problem they are solving. [12]. The underlying principles of GA were first published by Holland in 1962, [13]. The mathematical framework was developed in the 1960s and is presented in his pioneering book in 1975 [14]. In optimization applications, they have been used in many diverse fields such as function optimization, image processing, the traveling salesperson problem, system identification and control. A high-level description of GA has been done by Davis in 1991 as follows. [15] Given a way or a method of encoding solutions of problem into the form of chromosomes and given an evaluation function that returns a measurement of the cost value of the following steps:

Step1: Initialize a population of chromosomes

Step2: Evaluate each chromosomes in the population.

Step3: Create new chromosomes by mating current chromosomes; apply mutation and recombination as the parent chromosomes mate.

Step4: Delete members of the population to make room for new chromosomes.

Step5: Evaluate the new chromosomes and insert them into the population.

Step6: If the stopping criterion is satisfied, then stop and return the best chromosome; otherwise, go to step3

3.2 Edge detection by CNN and genetic algorithms

In this work, proper CNN templates are described to locate edge detection in image by using genetic algorithms. For this aim, CNN templates are designed so that they satisfy the stability. So, A and B templates are selected as symmetric. Because of selecting size of templates as 3×3 , totally 11 template parameters are searched. One of these parameters is offset, five of them belongs the A matrix, and the other five parameters belongs the B matrix. Each parameter are encoded by 16 bits in chromosomes. So, the length of chromosomes has been selected as 176 bits. In training process, 72 chromosomes are constructed as initial population randomly. The number of population is kept constant as 72 during the algorithm. Mutation probability m_p has been set % 1. Training process includes these steps as follows.

- (i). Construct initial population; A matrix is constructed called as population matrix. Each row of the population matrix represents chromosomes. Because of selecting number of chromosomes is 72, there are 72 rows in population matrix. Number of columns of this matrix is 176, because there are 176 bits in each chromosome. At the beginning this matrix is constructed randomly.
- (ii). Extract the CNN template: Chromosomes represents the binary codes of the elements of the CNN template A, B, I . In this step, each chromosomes are decoded the elements of the CNN are computed in $[-8,8]$ interval. Since each element is coded as 16 bits, each parameter can take 216 different value in $[-8,8]$ interval. In each chromo-

mosomes first 11 bits represents first bits of the template elements. And second 11 bits of chromosomes represents the second bits of the template elements and so on. These elements are

$$S = [A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}, A_{11}, B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10}, B_{11}] \quad (5)$$

(iv). Evaluate cost function value for each chromosomes; In this step, an image which was selected as training image is given as input to CNN. Normally in this gray-level image, brightness varies in 0 (black) through 1(white) interval. To fit this image to CNN operation, brightness of the image is converted from [0,1] to [-1,1]. According the same rule, brightness of the CNN output image is converted from [-1,1] to [0,1]. Then CNN works with templates belonging with first chromosome. After the CNN output appears as stable, cost function is computed between this output image and target image which we want to obtain. This process is repeated with template sets belongs each chromosomes in the population. Cost function has been selected in this study as follow

$$Cost(A, B, I) = \sum_i \sum_j P_{i,j} \oplus T_{i,j} \quad (6)$$

3.3 Time multiplexing simulation approach

In this procedure it is possible to define a block of CNN processors which will process a subimage whose number of pixels is equal to the number of CNN processors in the block. The processing within this subimage follows the raster approach adapted in Chua and Yang (1988b). Once convergence is achieved, a new subimage is processed. The same approach is being carried out until the whole image has been scanned. It is clear that with this approach the hardware implementation becomes feasible since the number of CNN processors is finite. Also, the entire image is scanned only once since each block is allowed to fully converge. An important point is to be noticed that the processed border pixels in each subimage may have incorrect values since they are processed without neighboring information only local interactions are important for the latency of CNNs. To overcome the aforementioned problem two sufficient conditions must be considered while performing time-multiplexing simulation. Alternatively, to ensure that each border cell properly interacts with its neighbors it is necessary to have the following. (1) To have a belt of pixels from the original image around the subimage and (2) to have pixel overlaps between adjacent subimages.

It is possible to quantize the processing error of any border cell C_{ij} with neighborhood radius of 1. By computing independently the error owing to the feed forward operator and interaction among cells for the two horizontally adjacent processing blocks, the absolute processing error owed only to the effect of the B template is obtained by subtracting the erroneous state value from the error free states using

4 Numerical Integration Methods

four of the single-step numerical integration algorithms used in the CNN behavioral simulator described here. They are Euler, RK4(2), RK4(3) and RK8(6).

4.1 The Proposed Methods

R.Ponalagusami and S.Senthilkumar introduced Time-Multiplexing CNN using Limiting Formula RK (7,8) (R.Ponalagusami and S.Senthilkumar 2008). In this paper, we consider the same problem but by using different approaches RK4(2), RK4(3) and RK8(6).

4.1.1 Stepsize selection algorithm.

There are currently two widely used methods that have appeared in the literature for changing the stepsize of p (q)-order RK codes. The first is to apply the formula (see [9])

$$h_{n+1} = f_1 \left(\frac{TOL}{EST_n} \right)^{\frac{1}{p}} \quad (10)$$

Where f_1 is a safety factor and the new sought-after stepsize $h_{n+1} = x_{n+1} - x_n$ is predicted in terms of an estimate of the local error EST_n which is based on the approximation

$$EST_n \approx y_n - y_{n^*}$$

Assuming y_n, y_{n^*} to be the pth-, qth-order approximate solutions, respectively, at the previous grid point x_n and TOL the requested tolerance. If

$$EST_n \leq TOL,$$

Then the computed solution y_{n+1} is accepted and the integration is carried out, otherwise (5) is reevaluated by substituting

$$EST_n \rightarrow EST_{n+1}$$

This methodology is termed the error per step (EPS) mode (see Shampine [10]).

An alternative is to consider the same algorithm (5), but to use, instead of (6), the approximation

$$EST_n \approx \frac{y_n - y_{n^*}}{h_n}$$

This is called error per unit step (EPUS) [10].

4.1.2 RK4(2) and RK4(3) at n = 4

According to [8], The equations of RK4(2) and RK4(3) are:

$$\begin{aligned} k_{1j} &= \tau f'(x_{ij}(nr)) \\ k_{2j} &= \tau f'(x_{ij}(nr)) + \frac{5}{14} k_{1j} \\ k_{3j} &= \tau f'(x_{ij}(nr)) - \frac{52}{605} k_{1j} + \frac{819}{1210} k_{2j} \\ k_{4j} &= \tau f'(x_{ij}(nr)) + \frac{2576}{4745} k_{1j} - \frac{252}{365} k_{2j} + \frac{1089}{949} k_{3j} \\ k_{5j} &= \tau f'(x_{ij}(nr)) + \frac{19}{130} k_{1j} + \frac{343}{1215} k_{2j} + \frac{1331}{3159} k_{3j} \\ &+ \frac{73}{486} k_{4j} \end{aligned} \quad (1)$$

Therefore, the final integration is a weighted sum of the five calculated derivatives is given:

$$\begin{aligned} x_{ij}((n+1)\tau) &= x_{ij}(nr) + \frac{19}{130} k_{1j} + \frac{343}{1215} k_{2j} + \frac{1331}{3159} k_{3j} \\ &+ \frac{73}{486} k_{4j} \end{aligned} \quad (12)$$

The difference between RK4(2) and RK4(3) is the local truncation error in the case of RK4(2) is given by using the RK(2)i.e.

$$y_{ij}((n+1)\tau) = y_{ij}(nr) + \frac{4}{55} k_{1j} + \frac{343}{1215} k_{2j} + \frac{13}{486} k_{3j} \quad (13)$$

But local truncation error in the case of RK4(3) is given by using the RK(3)i.e.

$$y_{ij}((n+1)\tau) = y_{ij}(nr) + \frac{11}{130} k_{1j} + \frac{637}{1215} k_{2j} + \frac{605}{3159} k_{3j} - \frac{73}{1215} k_{4j} \quad (14)$$

5 Conclusion

As researchers are coming up with more and more CNN applications, an efficient and powerful simulator is needed. So we use simulated annealing in optimizing CNN using the numerical integrations, especially using RK4(2), RK4(3) and RK8(6) for more efficiency with genetic algorithms. The

simulator hereby presented meets the need in six ways: 1) Depending on the accuracy required for the simulation, the user can choose from three numerical methods to perform the numerical integration, 2) The input image format is JPEG, which is commonly available, 3) The input image can be of any size, allowing simulation of images available in common practices, 4) CPU time of our methods is better than those in the literature, 5), the quality measures of the pictures and the edge detection for our method is better than those in the literature.

REFERENCE

- [1] R.Ponalagusami and S.Senthilkumar(2008). "Time-Multiplexing CNN Simulation using Limiting Formula RK(7,8)".Research Journal of Information Technology 1(1):1-16. | [2] L. O. Chua and L. Yang(1988). "Cellular Neural Networks: Theory & Applications," IEEE Trans. Circuits and Systems, Vol. CAS-35, pp. 1257-1290. | [3] L.O. Chua and T. Roska(1992). "The CNN Universal Machine Part 1: The Architecture", in Int. Workshop on Cellular Neural Networks and their Applications (CNNA), pp. 1-10. | [4] J. A. Nossek, G. Seiler, T. Roska and L. O. Chua (1992). "Cellular Neural Networks: Theory and Circuit Design," International Journal of Circuit Theory and Applications, Vol. 20, pp. 533-553. | [5] J. Varrientos and E. Sanchez-Sinencio(1992), "CELLSIM: A cellular neural network simulator for the personal computer," in Proc. 35th Midwest Symp. Circuits Systs, pp. 1384-1387. | [6] W. H. Press, B. P. Flannery, S.A. Teukolsky, and W.T.g Vetterling(1986). "Numerical Recipes. The Art of Scientific Computing", Cambridge University Press, New York. | [7] P.J.M. van Laarhoven and E.H.L. Aarts, (1987) Simulated Annealing: Theory and Application. ISBN 90-277-2513-6 | [8] Ch. Tsitouras and S. N. Papakostas(1991) "Cheap Error methods for Runge-Kutta methods ", SIAM J. SCI. COMPUT, Society for Industrial and Applied Mathematics, Vol. 20, No. 6, pp. 2067-2088. | [9] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick(1972), Comparing numerical methods for ordinary differential equations, SIAM J. Numer. Anal., 9, pp. 603{637. | [10] L. F. Shampine(1986), Some practical Runge-Kutta formulas, Math. Comp., 46, pp. 135{150. | [11] T. Sziranyi, M. Csapodi, " Texture Classification and Segmentation by Cellular Neural Network Using Genetic Learning", Research Report ,Budapest,Hungary, November,1994 | [12] C.T. Lin, C.S. George Lee " Neural Fuzzy Systems " , Prentice-Hall Inc., New Jersey, 1995. | [13] J.H. Holland, "Outline for a logical theory of adaptive systems.", J. Assoc. Computing. Mach. 3:297-314, | [14] J.H. Holland, "Adaptation in neural and artificial systems", Ann Arbor, MI: University of the Michigan Press, 1975 | [15] L. Davis, "Handbook of Genetic Algorithms" New York:Van Nostrand, Reinhold,1991. | 1988 | [16] T. Kozek, T. Roska, L.O. Chua, "Genetic Algorithms for CNN template Learning", IEEE Trans. On Circuit and Systems, Vol.40, No.6 pp. 392-402, 1988 | [17] T. Sziranyi, M. Csapodi, " Texture Classification and Segmentation by Cellular Neural Network Using Genetic Learning", Research Report ,Budapest,Hungary, | November,1994 | [18] J.Hu, H.Yan, M. Sakalli, "Locating head and face boundaries for head-shoulder images", Pattern Recognition 32 (1999) 1317- | 1333 | [19] A. Samal, P.A. Iyengar, "Automatic recognition and analysis of human faces and facial expressions : a survey", Pattern Recognition 25 (1) (1992) 65-77 | [20] R. Chellappa, C.L. Wilson, S. Sirohey, "Human and machine recognition faces: a survey", Proc. IEEE 83 (5) (1995) 705-740. | [21] S.H. Jeng, H.Y.M. Liao, C.C.Han, M.Y. Chern, Y.T.Liu "Facial feature dedection using | geometrical face model: an efficient approach" | [22] C.T. Lin, C.S. George Lee " Neural Fuzzy Systems " , Prentice-Hall Inc., New Jersey, | 1995.