Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 6, No. 3, 2023

Learning for Cellular Neural Networks Using the GARPLA Algorithm

Duong Duc Anh [1], Pham Duy Tuan [2], Nguyen Quang Hoan [2], Nguyen Hong Vu [1],

Doan Hong Quang [3] and Lai Thi Van Quyen [1]

[1] Vietnam Research Institute of Electronics, Informatics and Automation, Hanoi, Vietnam
[2] Posts and Telecommunications Institute of Technology, Hanoi, Vietnam
[3] National Center for Technological Progress, Hanoi, Vietnam

Abstract: The main content of this article is to build a hybrid algorithm between genetic algorithm and Recurrent Perceptron Learning Algorithm to determine the weight set for standard cellular neural networks. In particular, the Recurrent Perceptron Learning Algorithm was announced by Gukzelis in 1998 to determine the weight set for CeNNs, however, the nature of the algorithm built according to LMS learning rules means there is still a possibility of local convergence problems of algorithm. Meanwhile, the genetic algorithm originates from natural selection problems and is capable of determining the local optimal domain of the problem. Therefore, the GARPLA hybrid algorithm will fully determine the optimal weight set for the CeNNs network, reducing potential local optimization problems encountered in the RPLA algorithm.

Keywords: Edge Detection; Recurrent Perceptron Learning Algorithm (RPLA); Genetic Algorithm (GA); Weight; Cellular Neural Networks (CeNNs).

1. Introduction

Currently, the application of neural networks to image processing problems such as edge detection and separation [1], [2] is becoming increasingly popular. Convolutional neural networks are derived from multi-layer Perceptron networks [3], which have been matured and are available in commercial versions. Another current research direction is to use recurrent neural networks to embed integrated circuits for specific image processing problems [4]. Cellular neural networks are one of the typical structures of recurrent neural networks capable of high-speed image processing [5], [6] and embedded in integrated hardware circuits such as FPGA and ASICS [7]. Therefore, research and development of content related to cellular neural networks needs to be promoted and implemented in the current context.

Because the cellular neural network belongs to the recurrent network class, the weight determination is usually used according to Hebb's law and only determines the response weight **A**. Thus, the influence of the **B** and **I** weight matrices are not considered to the network structure. In 1998, Gukzelis [8] proposed an RPLA algorithm to fully determine the weight set for CeNNs. However, the origin of the algorithm comes from the LMS method, so there is a possibility of a local extrema problem with the algorithm results. In order to minimize this case, we build a hybrid algorithm between genetic algorithm and RPLA to determine the global optimal value for weight matrices for CeNNs network.

2. Methodology

2.1. Stucture of Celullar Neural Networks

CeNNs are built and developed based on the architecture of an analog circuit [9]. Each circuit of CeNNs is considered a cell, which contains linear components such as capacitor *C*, resistor R_x and nonlinear components including nonlinear control sources and external sources.

The system of dynamic equations of basic CeNNs is as follows:

$$C\frac{dx_{ij}(t)}{dt} = -\frac{1}{R_x}x_{ij}(t) + \sum_{(k,l)\in N_r(i,j)} \mathbf{A1}(i,j;k,l)\, y_{kl}(t) + \sum_{(k,l)\in N_r(i,j)} \mathbf{B1}(i,j;k,l)\, u_{kl} + I \quad (1)$$

$$0 \le i,k \le M; 0 \le j,l \le N$$

Output interaction function:

$$y_{ij}(t) = \frac{1}{2}(|x_{ij}(t)+1| - |x_{ij}(t)-1|) \quad (2)$$

Constraints for stable CeNNs:

$$\mathbf{A1}(i, j; k, l) = \mathbf{A1}(k, l; i, j) \quad (3)$$

Assumed conditions

$$u_{ij} \le 1 \; ; \; x_{ij}(0) \le 1 \quad (4)$$

In there:

r: neighborhood radius of cell C(i, j). This problem uses the neighborhood radius *r = 1*, then *C(i, j)* is affected by 08 neighboring cells and the feedback of *C(i, j)* itself.

C, R_x: capacitor, resistance of CeNNs

$\mathbf{A1}(i,j;k,l), \mathbf{B1}(\boldsymbol{i,j;k,l})$: weight matrix describes the connection between cell *C(i, j)* with neighboring cells *C(k, l)* of the output and input signals respectively; size (3x3).

(i, j), (k, l): represents the position of the cell in CeNNs of size (MxN).

M, N: number of cells of CeNNs horizontally and vertically, respectively.

I: threshold matrix, size (1x1).

$x_{ij}(t) \in R^n$: cell state (i, j), size (MxN).

$y_{ij}(t) \in R^n$: network output signal matrix; size (MxN).

$u_{ij} \in R^n$: input signal of the network; size (MxN).

Thus, the network structure outlined in equation (1) is depicted in the following form:

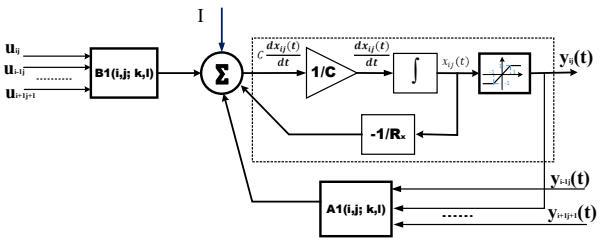


Figure 1. Structure diagram of the CeNNs

2.2. Hybrid Algorithm GA combined with RPLA (GARPLA)

As demonstrated in the preceding text, the Perceptron algorithm emerges as a distinct instance of LMS learning rules, indicating a propensity for local convergence in its outcomes. This document introduces an algorithm designed to secure global optimization, incorporating a fusion of GA and RPLA algorithm. The GA assumes a pivotal role in delineating the global optimal value domain, while the RPLA algorithm guarantees the identification of the algorithm's extreme point. The details of this approach are outlined below:

To implement the hybrid algorithm, we divide it into two steps:

Step 1: Use GA to calculate a set of weights for CeNNs. Then, this set of weights is guaranteed to be in the globally convergent value range of the weight sets.

Step 2: From the weight sets calculated from GA, select the initial weight matrix for the RPLA algorithm. Use the RPLA algorithm to calculate the globally optimal weight set for CeNNs.

2.2.1. The GA for Cellular Neural Networks

The genetic algorithm is a prominent method in evolutionary computing extensively used for addressing optimization challenges, particularly notable for its application in the field of neural networks [10], [11], [12]. This article presents the application of genetic algorithms within the framework of cellular neural networks. The algorithm is employed to calculate the initial set of weights, including **A1, B1** and **I** that is the input for the next learning phase.

In the content of the article, we want to use a cellular neural network for the problem of determining object image boundaries. Therefore, besides being based on the input learning sample set, we need to choose appropriate parameters of the GA for the cellular neural network including the following:

Gene: A base-10 gene, representing the digits in each chromosome, is chosen for our selection process. Given that the computation of Cellular Neural Networks (CeNNs) involves a continuous optimization process, the adoption of the base-10 gene is particularly fitting. The utilization of these decadal genes has the capacity to enhance diversity within the population, contributing to a more comprehensive exploration of the solution space.

Chromosomes: includes all operands in the set of weight matrices **A1, B1, I**. However, according to the condition in equation (3), the matrix **A1** is symmetric, so only five values (chromosomes) need to be determined. Meanwhile, we consider matrix **B1** to be symmetrical similar to **A1** for ease of calculation, so we only need to determine five weights (chromosomes) and one chromosome of **I**. Thus, there are a total of 11 chromosomes that need to be symmetrically determined with cellular neural networks. Here, we use the limit value of chromosomes from: -99.99 to +99.99

Population: Includes 11 parent populations corresponding to 11 chromosomes whose weights need to be determined. In particular,each parental population includes 5 father individuals and 5 mother individuals to serve the Crossover process.

Fitness function: We use the fitting function which is the least squared error function. Because the purpose of the learning process is to determine the object boundary in the image, when the sum of squared differences between the desired output image and the calculated output is less than an e_{min} value, the learning process ends.

Determine the state value, input and output of CeNNs.

The GACeNNs algorithm is presented specifically as follows:

Input:

i) The cellular neural network structure satisfies expressions (1) to (4)

ii) Deviation value $e_{1\min}$ for Genetic Algorithm

iii) Choose the initial set of Weights matrices:

$$\boldsymbol{A1[0]} = \begin{bmatrix} a_{11}[0] & a_{12}[0] & a_{13}[0] \\ a_{14}[0] & a_{15}[0] & a_{14}[0] \\ a_{13}[0] & a_{12}[0] & a_{11}[0] \end{bmatrix}, \quad \boldsymbol{B1[0]} = \begin{bmatrix} b_{11}[0] & b_{12}[0] & b_{13}[0] \\ b_{14}[0] & b_{15}[0] & b_{14}[0] \\ b_{13}[0] & b_{12}[0] & b_{11}[0] \end{bmatrix}, \quad \boldsymbol{I[0]} = I[0]$$

iv) Select 11 initial parent populations, with base 10 Genes, each parent individual has 5 Genes (including markers)

v) Choose the rate of hybridization and mutation

Output: Weight set of CeNNs

Step 1: Initialize

Initialize the initial conditions of the algorithm according to the data provided in Input.

Step 2: Evaluate the fitness function:

$$E = \frac{1}{2}\left(d_{ij}^{s} - y_{ij}^{s}\right)^2 \rightarrow E_{min} \tag{5}$$

Step 2.1: If $E > E_{min}$then go to step 3;

Step 2.2: If $E \leq E_{min}$ the weight value satisfies the problem. Go to step 6;

Step 3: Calculate the state value $x_{ij}(t)$, the calculation output $y_{ij}(t)$ of CeNNs.

Step 4: Crossover

Perform a crossover of each parent population for each weight of CeNNs, select the optimal chromosome in the new chromosome population generated during the breeding process by how to calculate the fitness function value generated by each new chromosome. The new chromosome that creates the smallest fitness function will be selected to replace the original chromosome of CeNNs. Each chromosome undergoes training following the principle of left-to-right, top-to-bottom progression, in the specified order from matrix **A1, B1**, up to **I**.

Step 4.1: When $E = \frac{1}{2}\left(d_{ij}^{s} - y_{ij}^{s}\right)^2 \leq e_{min}$moving to step 6;

Step 4.2: When $E = \frac{1}{2}\left(d_{ij}^{s} - y_{ij}^{s}\right)^2 > e_{min}$ moving to step 5;

Step 5: Perform mutation for some weights in the network. Return to step 2 to determine the weight matrix for the next rounds.

Step 6: Stop the algorithm

2.2.2. The RPLA for Cellular Neural Networks

The regression Perceptron algorithm is used to calculate the weights of standard CeNNs in steady state [13]. In 1994, C. Gukzelis proposed the RPLA algorithm by combining the response matrix **A1**, input matrix **B1**, and threshold matrix **I** into weight matrix **W** [14]. When standard CeNNs reach a

stable state then (1) has a left-hand side equal to 0. Then, (1) can be rewritten as follows:

$$\frac{1}{R_x}x_{ij}(t) = I + \sum_{(k,l)\in N_r(i,j)} \boldsymbol{A1}(i,j;k,l)\, y_{kl}(t) + \sum_{(k,l)\in N_r(i,j)} \boldsymbol{B1}(i,j;k,l)\, u_{kl} \quad (6)$$

Set **W** matrix:

$$\boldsymbol{W} = [\boldsymbol{A}^T\ \boldsymbol{B}^T\ \boldsymbol{I}] \quad (7)$$

And **Y** matrix:

$$\boldsymbol{Y}_{\text{ij}}(t) = [y_{kl}^T\ \text{u}_{kl}^T\ 1] \quad (8)$$

It is easy to see that when considering the chosen R_x as a standard unit, equation (3) in steady state is equivalent to the feedforward Perceptron network:

$$Ax_{\text{ij}}(\infty) = \left[\boldsymbol{Y}_{i,j}\right]^T * \boldsymbol{W} = \left[\left[y_{i,j}\right]^T \left[u_{i,j}\right]^T 1\right]^T * \boldsymbol{W} \quad (9)$$

So from equation (), the recurrent CeNNs structure has transformed into a 1-layer feedforward network. Then the network's learning rules will be used according to the trial-and-error method as follows:

$$\Delta w_{ij} = w_{ij}(n+1) - w(n) = -\alpha * (d_{ij}^s - y_{ij}^s(\infty))\boldsymbol{Y}_{ij}^+ \quad (10)$$

Trong đó:

d_{ij}^s: desired output signal

$y_{ij}^s(\infty)$: Actual output signal at steady state

α: learning rate

$\boldsymbol{Y}_{ij}^+$: The input to the network corresponds to each cell location

The sequence of steps of improved Perceptron learning is as follows:

Input: Given:

1) Set of CeNNs training samples including $((u_{\text{ij}}^s, x_{\text{ij}}^s(0)), d_{\text{ij}}^s)$,

2) Learning rate α.

3) Select *e_{2min} for RPLA*

4) Use the set of weight matrices calculated from GA.

Output: the parameter set of CeNNs

Step 1: Use a set of weight matrices calculated from the results of the GA algorithm

Step 2: Calculate the total difference between the output $y_{\text{ij}}^s(\infty)$and the known desired pattern d_{ij}^s

1. If the total error is less than the value *e_{2min}*, go to step 6

2. If the total error is more than the value *e_{2min}*, go to step 3

Step 3: Update the CeNNs' weight set;

Step 4: Calculate the state value $x_{\text{ij}}^s[n]$

Step 5: Calculate the output of the network $y_{\text{ij}}^s[n]$; return Step 2

Step 6: Finish.

3. Experiment

3.1. Problem Statement

Give: i) the structure of the CeNNs according to the equation (1) to (4);

ii) the network training templates $\left[(x_{ij}^s(0), u_{ij}^s), d_{ij}^s\right]$with size of *8*8*, 03 templates for learning algorithm follow Figure 2 below.

Need to detection the edge of image. To solve the problem, need to perform 02 phases: Learning phase and pattern recognition phase.

3.2. Learning Phase

3.2.1. Genetic Algorithm

Choose a network structure described by expressions (1) to (4)

Select the initial weight matrix set of CeNNs:

$$\boldsymbol{A1}[0] = \begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix},\quad \boldsymbol{B1}[0] = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix};\quad \boldsymbol{I}[0] = 1$$

Select minimum error $e_{1min} = 10$

Select the appropriate parameters Genes, Chromosomes, Populatiosn, and fitness function according to sections 2.2.1

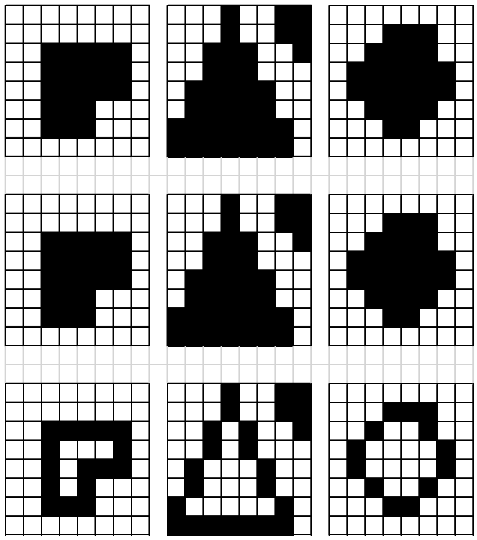


Figure 2. Templates for image processing

For each weight in the network, we follow the ordering principle from matrices **A1**, **B1**, and **I**. The calculation of all corresponding weights defines an epoch, with 11 chromosomes per epoch. Remarkably, after the 6th epoch, the total error converges to the desired optimal value. The calculated weight values constitute the sought-after set, concluding the weight calculation algorithm for SOCeNNs using the genetic algorithm. The resulting set of matrices is as follows:

$$\boldsymbol{A1}[6] = \begin{bmatrix}1.27 & -1.06 & -2.05\\ -4.2 & 6.98 & -4.2\\ -2.05 & -1.05 & 1.28\end{bmatrix},\quad \boldsymbol{B1}[6] = \begin{bmatrix}-2.83 & -3.14 & 0.7\\ 0.27 & 9.64 & 0.27\\ 0.7 & -3.14 & -2.83\end{bmatrix},\quad \boldsymbol{I}[6] = -4.55$$

3.2.2. Recurrent Perceptron Learning Algorithm

Leveraging the outcomes derived from the Genetic Algorithm (GA) and the training sample set outlined in the previous content, the learning process for the network unfolds through RPLA. The objective is to ascertain the globally optimal weight set for the cellular neural network. During the learning process using RPLA, we reduce the minimum error compared to the minimum error of GA, here choosing $e_{2min} = 4$, and select learning rate $\alpha = 0.01$. In the learning process employing RPLA, we achieve a set of matrices after 12 epochs. These weight matrices lead to the total error between the output and input of the three samples reverts to the desired minimum value.

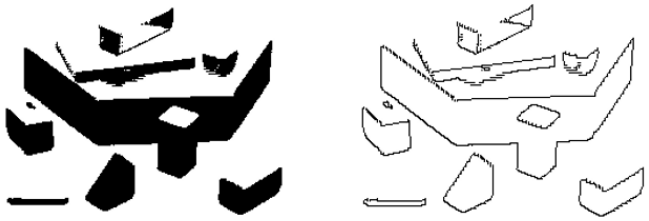
$$\boldsymbol{A1}[12] = \begin{bmatrix}1.13 & -1.24 & -2.13\\ -4.4 & -6.45 & -4.4\\ -2.13 & -1.05 & 1.13\end{bmatrix},\quad \boldsymbol{B1}[12] = \begin{bmatrix}-3.08 & -3.84 & 0.2\\ 0.18 & 8.56 & 0.18\\ 0.2 & -3.14 & -3.08\end{bmatrix},\quad \boldsymbol{I}[12] = -5.42$$

Using this ultimate set of weight matrices, the objective is to generate actual output matrices closely mirroring the desired output matrices. This signifies the successful completion of the edge detection problem through RPLA, achieving the desired level of accuracy.

3.3. Pattern Recognition Phase

Using the set of weights acquired through the hybrid algorithm of GA and RPLA, we conducted experiments on the image edge detection problem. Employing the PyCeNNs image processing algorithm developed for Cellular Neural Networks from an open-source code on Github, implemented in Python, we carried out two distinct testing phases as outlined below:

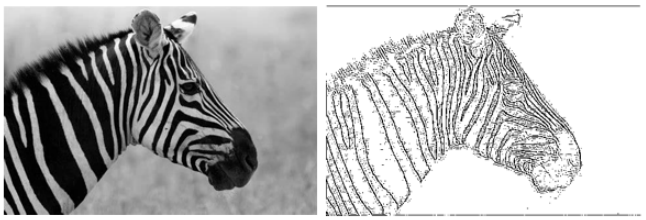
Case 1: In this scenario, a 256x256 input image previously processed by independent algorithms GA and RPLA to extract image edges is utilized. In the left image (Fig 3.a), diverse blocks are arranged without fixed principles. Conversely, Figure 3.b portrays the object after boundary separation. The outcomes from these algorithms describe the object boundaries are clearly defined, showcasing a smooth and uninterrupted appearance.



a. Input image b. Output image GARPLA

Figure 3. Result of edge detection in image processing

Case 2: Opting for a black and white image featuring a zebra head (Figure 4) with dimensions (474x316) as the input image, the image produced through the collaborative GA and RPLA algorithm presented the boundary of the zebra's head. It's essential to observe that, in this instance, the GARPLA algorithm does not effectively reveal detailed and distinct object boundaries in the image.



c. Input image d. Output image GARPLA

Figure 4. Result of edge detection in image processing

Through the above two test cases, using visual methods, we found that for small sized, less complex images, using CeNNs separation is completely appropriate. Meanwhile, for the image of a zebra head with large size and many complex details, CeNNs have not completely solved the problem of object boundary separation. In fact, we also conducted many different cases but all got the same result.

4. Summary

The article details the Perceptron regression algorithm and the genetic algorithm to clarify how to calculate the weight set for the cellular neural network. From there, build the GARPLA algorithm by hybridizing the above two algorithms. The advantage of this algorithm is that it helps us fully calculate the weights of CeNNs without local optimization occurring. Calculating weights according to this algorithm ensures CeNNs are stable and applicable to specific processing image problems. The algorithm was tested in image processing to determine the boundaries of different objects, and encouraging results were obtained.

From the results of the article, we find that CeNNs have suitable applications for image processing problems. However, the image matrix is large in size so the use of high-order CeNNs is necessary. The next development direction is to build weighting algorithms for high-order CeNNs. Our research team is conducting trials in new research directions that can be applied to more complex problems.

References

[1] N. Behar, M. Shrivastava, „A Novel Model for Breast Cancer Detection and Classification, “Engineering, Technology & Applied Science Research, Bd. 12, Nr. 6, pp. 9496-9502, 2022.

[2] A. Alsheikhy, Y. Said, M. Barr, „Logo Recognition with the Use of Deep Convolutional Neural Networks, “Engineering, Technology & Applied Science Research, Bd. 10, Nr. 5, pp. 6191-6194, 2020.

[3] G. S. Fesghandis, A. Pooya, M. Kazemi, Z. N. Azimi, Comparison of Multilayer Perceptron and Radial Basis Function Neural Networks in Predicting the Success of New Product Development, Engineering, Technology & Applied Science Research, Bd. 7, Nr. 1, pp. 1425-1428, 2017.

[4] L. Chua, T. Roska, Cellular Neural Networks and Visual Computing, Cambridge, United Kingdom: Cambridge University, 2004.

[5] M. Gabriele, „Another Look at Cellular Neural Networks, “in CNNA, Catania, Italy, 2021.

[6] S. Husain, M. Imran, A. Ahmad, Y. Ahmad, K. Elahi, „A Study of Cellular Neural Networks with Vertex-Edge Topological Descriptors, “Computers, Materials & Continua, Bd. 70, Nr. 2, pp. 3433-3447, 2022.

[7] E. Kose, Mus ¨¸tak E. Yalc¸ın, „A New Architecture for Emulating CNN with Template Learning on FPGA, “in CNNA, Budapest, Hungary, 2018.

[8] Gukzelis,C.; Karamamut, S.; and Genc,I .(1999). A Recurrent Perceptron Learning Algorithm for Cellular Neural Networks, Springer-Verlag, Bd. 51 (1999), pp. 296-309.

[9] L. Chua, L. Yang, „Cellular Neural Networks: Theory,“ IEEE Transactions on Circuits and Systems , Bd. 35, Nr. 10, pp. 1257-1272, 1988.

[10] T. Kozek, T. Roska, Leon 0. Chua, „Genetic Algorithm for CNN Template Learning, “IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-, Bd. 40, Nr. 6, pp. 392-402, 1993.

[11] E. Gomez-Ramirez, X. Vilasis-Cardona, „Cellular Neural Networks Learning using Genetic Algorithm,“ Revista del Centro de Investigación de la Universidad la Salle, Bd. 6, Nr. 21, pp. 25-31, 2003.

[12] R. V. V. Krishna, S. Srinivas Kumar, „Hybridizing Differential Evolution with a Genetic Algorithm for Color Image Segmentation, “Engineering, Technology & Applied Science Research, Bd. 6, Nr. 5, pp. 1182-1186, 2016.

[13] A. Slavova, Cellular Neural Networks: Dynamics and Modelling, Sofia. Bulgaria: Kluwer Academic, 2003.

[14] C. Guzelis; S. Karamahmut, „Recurrent Perceptron Learning Algorithm for Completely Stable Cellular Neural Networks,“in The Third IEEE International Workshop on Cellular Neural Networks and their Applications, Rome, Italy, 1994.