International Journal of Computer Engineering and Information Technology

VOL. 9, NO. 5, May 2017, 87–96
Available online at: www.ijceit.org
E-ISSN 2412-8856 (Online)

Enhancing the Performance of CNNs using Evolutionary Programming

Abdelghafar M. Elhady[1], Elsayed Radwan[2], Hazem M. El-bakry[3] and Ahmed Abou Elfetouh[4]

[1, 3, 4] Faculty of Computer Sciences and Information Systems, Mansoura University, Egypt

[1, 2] Deanship of Scientific Research, Umm Al-Qura University, KSA

[1]abdelghafar.elhady@gmail.com, [2]elsfradwan@yahoo.com, [3]helbakry5@yahoo.com, [4]elfetouh@gmail.com

ABSTRACT

So far supervised learning of a dynamical system (Cellular Neural Network) is derived from a mathematical method that may lose some factors, like disturbance and noise data. This paper points to a new technique based on Evolutionary programming Computation (Genetic Programming) in order to overcome this problem. Because of the short in the prior techniques in counting the implementation problem, the hardware implementation constraints are taken into account when evolving new learning rules. Genetic programming is developed to discover the optimum supervised rules parameters and the form of the optimistic rule structure. With these new rules, the error rate and the learning time are reduced. The overall performance of CNNs is enhanced. Simulation results confirm the theoretical considerations. Moreover, comparisons with related work are given.

Keywords: *Genetic Programming, Cellular Neural Networks, Parametric Learning Rules, Hardware Implementation Constraints, Multi-Objective Fitness Function.*

1. INTRODUCTION

CNN is a dynamical system that combines the best characteristics of Cellular Automata [1] and Neural Networks [2, 3, 4]. The learning mechanisms in Cellular Neural Networks are usually accompanied with changes in the template weights (A, B, z) according a local/dynamical rule set. In such a model, learning rules control the variation of the parameters (the feedback template, the feedforward weight, and the threshold) of the CNN. Some priory researches in CNN have proposed learning methods based on mathematical principles [5, 6, 7]. These achievements, like the truncation learning, are totally built upon mathematical analysis which loss the data disturbance analysis (sensitivity analysis) wherever TL did not count the hardware implementation. Also, a new supervised learning rule has been discovered for uncoupled CNN using evolutionary programming, Genetic Programming (GP), which is a heuristic based method [8]. This technique loss representing the hardware implementation constraints and representing the coupled dynamical system of CNN. Moreover for any dynamical systems, the supervised learning are suffering from different problems [9]. The learning process is particularly slowness, and the training performance is sensitive to the initial conditions. During the learning process, oscillation may be yielded because the learning rate is high. Also, if the error function is shallow, the slope will be minor resulting in small template changes. Thus, the best learning mechanism should be achieved by simulating the human-like performance in satisfactory time.

By this paper, CNN (case study) is a 2D grid layout of simple processing cells locally adjacent [10]. This behavior makes CNN very well appropriate for secure communications, modeling wide area monitoring system (complex system) and hardware implementation. Moreover, because of the parallel processing abilities, CNN is suitable for those applications demanding high processing speed, like SCADA system [3]. The CNN dynamical system is complex to have supervised learning rules that control its parameter variations. So, this paper tries to achieve this task by heuristic techniques depending on the multiple evolutionary programming techniques that are evolved synchronously [11].

This paper aims to find new supervised learning rules that take into account hardware implementation constraints using a standard optimization programming technique, Genetic Programming (GP) [12, 13, 14]. GP is an

evolutionary computing technique similar to the Genetic Algorithms excepting its tree structure. This method assumes the supervised learning rules of the CNN as three parametric functions associated with the inputs and outputs, whose values differ for each synapse. Coupled to a scalar parameter (search bias z), which are the same for all synapses in the CNN, is also associated. By this work, Genetic Programming is suggested to overcome some of the learning problems and promise right operation of analogic CNN by means of introducing multi-objective fitness function. This technique saves the computational cost of the searching process for the admissible supervised learning rules since the search space concentrate on the implementation requirements.
In this paper, new supervised learning rules are discovered. In contrast with Truncation Learning Technique (TL), a comparative study on the learning time, number of epochs, and cumulative error rate among CNN cells has been illustrated. By This study, admissible supervised learning algorithm is appeared better than, but similar to, the TL rules and that GP is able to discover it. As a result of the previous discussion this paper is organized as; in section 2 preliminaries for Genetic Programming and Cellular Neural Network are given. In section 3 the evolution of the CNN learning rules with GP is discussed as well as the algorithm that describe the method is shown. Simulation results are given in section 4 and finally conclusions and future work are appeared as section 5.

2. PRELIMINARIES

2.1 Genetic Programming

Genetic Programming, GP[12], is an extension of the Evolutionary computing techniques, like GA, except in its structure. Wherever GA depends on a deterministic chromosome structure with approximate solution, GP depends on tree structure, varied in its size, with variable and operation representation. Though GP is used in reproducing a computer programs by undergoing adaptation. Unfortunately, since GP does not have a schema representation that describes its solution, GP should be fully described with hard constraints in its tree size. Furthermore, GP architecture should be completely described by specific leaf nodes (variable nodes), internal nodes (program operation) and evaluation function (fitness function). In addition, controlling parameters suitable for the problem constraints must be found. Moreover, a method for designing the results and criterion for terminating the run should be existed.
GP is a simulation for Darwin's Theory (evolutionary Theory) in reproducing new cells from others during time series. GP is developed to reproduce a new solutions based on exiting solutions in the current generation. GP imitates the genetic operation like recombination, mutation and reproduction. The recombination operator of GP depends on yielding a new solution from two other solutions under the name of crossover operation. Though crossover operation is a function $C: ST_i(A) \times ST_i(A) \rightarrow ST_{i+1}(A)$, where $ST_i(A)$ is the pole mate of all program trees defined at the current generation of the problem Universe A. The crossover drives two program trees in the current generation and reproduces a new offspring consisting of taken parts of each parent with different size and architecture. The offspring is inserted into the next generation $ST_{i+1}(A)$ as illustrated in Fig. 1. Also, there is another operation called Mutation operator $M: ST_i(A) \rightarrow ST_{i+1}(A)$. It creates a new child tree T' from an existing parent tree T, where $M(T) \mapsto T'$. Mutation randomly substitutes a node-tree with a varied created one reproducing a new offspring to be inserted into the next generation, as illustrated in Fig. 2. Finally, reproduction operator is a mapping $RE: ST_i(A) \rightarrow ST_{i+1}(A)$, where a selected individual tree is imitated for inclusion in the next generation, $RE(T) \mapsto T$. Because of genetic operators, the next generation is constructed by the current population, wherever the new population replaces the old one. The process of reproducing new generation is continuously repeated until a predefined number of generations reached or reach a solution with maximum standardized fitness function. The Algorithm of GP can be found by Koza J. in [12].

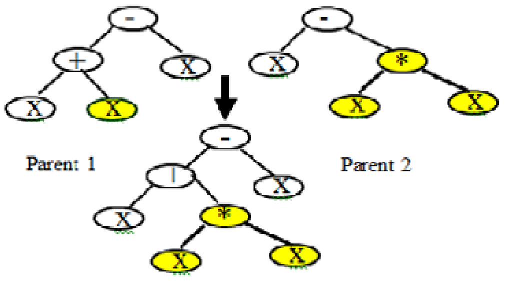


Fig. 1 .rossover operator.

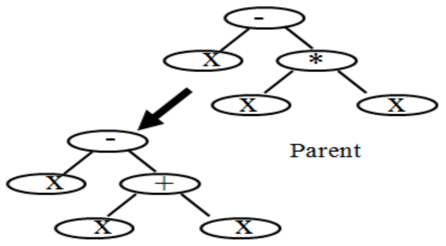


Fig. 2. Mutation operator.

2.1 Cellular Neural Networks

So far a fully connected Hopfield net is constructed for imitate the human brain. Unfortunately, Hopfield network is suffering from the overfitting problem. Thus, Hopfield network is difficult to be learned and need a lot of time to be processed and developed. Also, it is really a pipe dream to be implemented in hardware for a significant number of nodes. Though, Chua L. [15] solves these problems based on analogic parallel processors arranged in an $M \times N$ array. The suggested dynamical system offer a compromise by only connecting cells to their neighbors nearby. This dynamical system is called Cellular Neural Network, CNN. CNN is a grid layout of arranged cells that are analog dynamical processors adjacent locally[16]. CNN cells are only connected to other cells within a sphere of influence of radius r (N_r) and described by:

- **State equation:**

$$\frac{dx_{ij}}{dt} = -x_{ij} + z + \sum_{C_{kl} \in N_r(ij)} \hat{A}_{ij;kl}(y_{kl}(t), y_{ij}(t)) + \sum_{C_{kl} \in N_r(ij)} \hat{B}_{ij;kl}(u_{kl}(t), u_{ij}(t)) \quad (1)$$

where $\hat{A}$, $\hat{B}$ are the corresponding feedback and feedforward synaptic weights, respectively. x_{ij} , u_{ij} , y_{ij} are the state, input, and output voltage of the predefined CNN node, respectively. Whereas z is the network threshold.

- **Output equation:**

a strictly monotone increasing sigmoid function with tangent one in the interval $(-1,1)$ otherwise the slop is zero, whereas the CNN stability criteria , piecewise linear output function:

$$y_{ij} = f(x_{ij}(t)) = 0.5(|x_{ij} + 1| - |x_{ij} - 1|),$$

(2)

It can be converged by a differentiable smooth (C^1) function.

- **Input equation:**

$$u_{ij} = E_{ij} \quad (3)$$

- **Constraint equations:**

The boundary condition is considered constant by this work where.

$$|x_{ij}(0)| \le 1, \quad |u_{ij}(t)| \le 1$$

(4)

The subscript ij denotes to a grid node accompanied with a cell on the CNN array, and $kl \in N_r(ij)$ is a grid node in the neighborhood within the radius r of the cell ij. The CNN dynamic can be understood as sets of operational templates, A and B, and the scalar z. The cell value is reliant on the operations performed on the neighboring cells. CNN is analog-digital and continuous in time. It is a true parallel processor, resulting in very high speed.

3. EVOLUTION OF THE CNN LEARNING RULES WITH GP

GP is an experimental method that depends on evolutionary computing in its algorithm and able to search the whole space based on its tree structure and Genetic operations. Though GP is successfully applied in finding a general machine for automatic analog algorithm design independent of the solved problem [17,18]. So far GP has been used successfully in learning the template function for the centralized CNN [19,20]. In this paper, GP is used to find new supervised learning rules which have an optimal number of parameters depending on evolving three different trees simultaneously. The program tree structure is considered as numerous trees architecture. Each tree contains a function (program) that has evolved for a specified purpose. This tree structure is evolved for developing a supervised learning rule corresponding to the feedforward, feedback and network bias based on crossover, or other genetic operations. Complete programs are constructed at the same time each of its trees is improved as a result of multi-objective fitness function. The genetic operations handle these multi-trees structures one by one. While there are many different ways for evolving these architectures [13,21], the genetic operations are introduced to act upon only one tree at a time. Others remain unchanged and are transferred from the current population to next generation as an offspring. Genetic operations are restricted to a single tree at a time. This will save the building blocks of useful program (function).

By this method, each tree has different primitives (leaf nodes) corresponding to its program than others have. The leaf nodes (primitives) of the first tree are assumed corresponding to the feed-forward supervised learning rule. It is considered to be a function in the input and the parameters, the error which equal to the modulus of difference between the CNN settled output and the desired output and the first order derivative of the activation function. Since the output function, linearly sigmoid "piecewise" function of the state x_{ij} as demonstrated in Eq. (2), it can be converged by a smooth (C^1) sigmoid function. This function can be well converged with the classical tangent hyperbolic function. Hence, $y_{ij}(x)$ is defined as Eq. (5) and Eq. (6):

$$y_{ij} = f(x_{ij}) = \tanh(x_{ij})$$

(5)

$$f'(x_{ij}) = \sec h^2(x_{ij}) = 1 - \tanh^2(x_{ij}) = 1 - y_{ij}^2 \quad (6)$$

The leaf nodes (primitives) of the second tree associated with the supervised learning rule of feed-back template are output and the parameters, the error and the first order derivative of the activation function. Finally, the last tree is considered in accordance with the learning rule of the threshold. Since z is constant for all the CNN, it is considered to be a function in the parameters only, the error and the first order derivative of the activation function. The Network bias depends only on the cell itself.
For the Genetic Programming method the standard arithmetic operators (+,-,*) are used. Each time the three trees are integrated into the transient equation, where the transient equation is governed by the template built from the new supervised learning rules that are produced by the program trees. The standardized fitness function of the each program trees is the sum, taken over all CNN cells, of the quadratic value of the error.

$$g(p^r) = \sum_{i=1}^{S}(y_i^d - y_i(\infty))^2 = \sum_{i=1}^{M}\sum_{j=1}^{N}(y_{ij}^d - y_{ij}(\infty))^2 \qquad (7)$$

Where the pattern p^r denotes the template parameter vector, i.e. the best learned template (A, B, z) by the program-trees r. $S = M \times N$ is the array dimension of CNN. y_i^d is the desired output value of i^{th}cell and $y_i(\infty)$ is the settled output of the corresponding cell i. $g(p^r) = 0$ (error free) then template p^r outcome is matching to the desired output. The error function increases quadratically when the distance is greater than one and approximately zero elsewhere. The optimization problem can be formulated by means of the cost function but the hardware implementation constraints are missing. Since a single experiment is not expressible for all CNN templates, then the experiment is defined as linear combination among different experiments' result, where each experiment expresses on one of the CNN different template structures. So the modified raw fitness function is defined to be;

$$G(p_1{}^r, p_2{}^r, ..., p_m{}^r) = \sum_{c=1}^{m} w_c g(p^r{}_c), \quad \sum_{c=1}^{m} w_c = 1 \text{ and } w_i \in (0,1) \qquad (8)$$

where m is the total number of experiments, $p^r{}_c$ is the template corresponding to the experiment c based on the program-trees r, and w_c is the weight parameter which is calculated by $w_c = \frac{S_c}{S}$, S_c is the image size corresponding to the template c and $S = \sum_{c=1}^{m} S_c$ is the summation of all experiment sizes. GP uses $G(.)$ to be minimized. Indirectly, the raw fitness value of the program trees r is mapped into a standardized fitness value $f^r(.)$ to be maximized, as illustrated in Fig. 3.

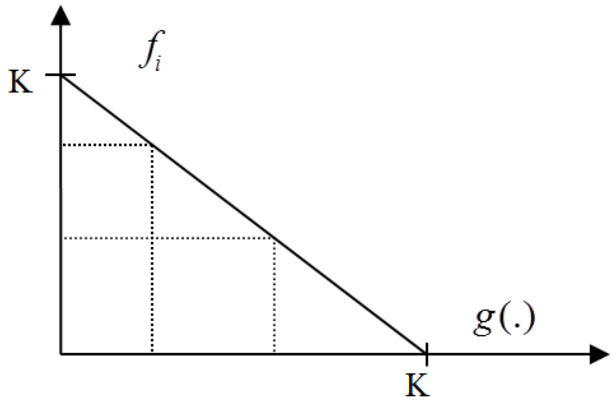


Fig. 3. The standardized fitness.

$$f^r = K - G(p_1^r, p_2^r, ..., p_m^r) \qquad (9)$$

where K is the supremum value taken over CNN cell output values, for image data $K = 4\sum_{c=1}^{m} w_c S_c$ and S_c is the image size corresponding to the template c [22]. By the CNN template parameters, The computation of the CNN output is stationary if the CNN reaches an equilibrium point or after predefined number of epochs. Also, the learning by the program trees is stopped if a template with zero error $G(P^r) = 0$, $f^r = K$, is appeared or after predefined number of learning epochs.

Because of the hardware implementation constraints, the CNN-type construction needs a confident degree of robustness as the result of the mismatching effects. Mismatch is the process that causes time independent random variations in physical quantities of identically designed devices. The mismatch can be decreased by increasing the chip surface which is not possible. Therefore, the optimal technique to reduce mismatch effect is the sensitivity analysis, where the variation in the input pattern is struggled by confidence interval (robust degree) in the template parameters. Naturally, a relative robustness degree against deviations of the token values (5-10 percentages) is sufficient to struggle the mismatch effect in the CNN chip. Thus, the relative fitness of a CNN single layer is defined as [23,24]:

$$D(p^r) = \max_{\alpha}\{\alpha \mid y_\infty(p^r \circ (1 + \alpha 1^{\pm})) = y_\infty(p^r) \text{ for all } 1^{\pm} \in \beta^j\} \qquad (10)$$

where $\circ$ denotes the component wise multiplication, $y_\infty(p^r)$ is the CNN settle output corresponding to the template p^r, $\beta = \{-1, 1\}$ and $j = (2r+1)(2r+1)+1$. Thus, the legal permutations of each CNN template should be examined 2^j for each value of α. Hence, for reducing the computation costs, the output is examined when the network has a stable value, $f^r = K - \varepsilon$ and $\varepsilon \ll 1$. Then, this constraint aims to guarantee solutions with respect to mismatching phenomena. For homogeneity with other different relative functions that represent the hardware constraints, a logarithmic function guarantees maximum penalty to those CNN templates with robustness under one percent as illustrated in Fig. 4. $g_2(p_i^r)$ is defined as multiple defined function with high penalty on the mismatching effect as equation 11.

$$g_2(p_i) = \begin{cases} (1-\log_{10}(D')) & 0.1\% \le D \le 10\% \\ 0 & D > 10\% \end{cases},$$

$$D' = 100D(p_i) \text{ and } D(p_i) = \min_{j\in\{1,\dots,m\}} \{D(p_{ij})\} \quad (11)$$

Since GP is able to search the whole universe and discovers the majority of the admissible supervised learning rules. The program result to be a program trees with maximum standardized fitness in addition to minimize $g_2(p_i)$ to guarantee hardware implementation constraints. Coupled to this, the number of learning epochs, high convergence speed, should be taken as a minimized function. Since the chosen starting template may be converge to the optimal one, i.e. the learning rules have a slight affection on the learning template, then we should assume strong punishment on the number of learning epochs less than a predefined number $N_{\min}$. The use of Gaussian function centered in $N_{\min}$ leads to a better performance, as illustrated in Fig. 5. The form of the constraint function for minimizing the number of epochs can be defined as;

$$g_3(p_i) = \begin{cases} 1-\exp(-\frac{abs(N_i - N_{\min})}{N_{\min}}) & if \;\; N_i \le N_{\min} \\ 1-\exp(-(\frac{abs(N_i - N_{\min})}{N_{\max} - N_{\min}})^2) & if \;\; N_i > N_{\min} \end{cases}$$

(12)

where $N_i = \max_{j\in\{1,2,\dots,m\}} \{N_{ij}\}$, N_{ij} is the number of learning epochs that is needed to get zero error for the experiment j in the i^{th} individual program, and $N_{\max}$ and $N_{\min}$ are the maximum and the minimum number of learning epochs assumed respectively. Then the problem of optimization, demonstrated in table 1, is defined to achieve the multi-objective fitness function;

$$\begin{cases} \max & f_i = K - G(p_1, p_2, \dots, p_m) \\ \min & \lambda g_2(p) + (1-\lambda) g_3(p) \end{cases} \quad (13)$$

where $\lambda \in [0,1]$. Since the priority for the admissible individual comes for the program with error free, we choose the program with max f_i to be the best individual in the generation. If there is more than one program has the same value, the best among the population, of f_i, the second constraint is chosen to search for the admissible one. Then the optimization process can be described by algorithm 1;

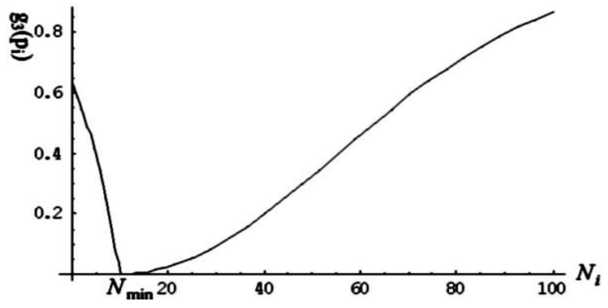


Fig. 4. The fitness value corresponding to the learning epochs.

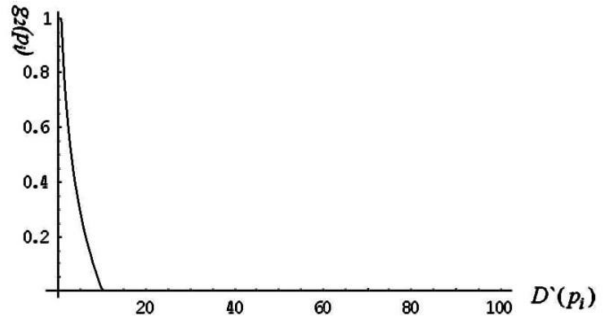


Fig. 5. The fitness value corresponding to the relative fitness.

Algorithm 1: The proposed method

INPUTS:

pop : Population size .
G : Maximum number of generations
E : Maximum number of Experiments
$N_{\max}$: The maximum times of learning epochs

OUTPUTS:

Output the admissible individual with the less structure complexity.

BEGIN

```
1. For n= 1 to pop step n+1 do
2. Generate random tree
3. Store generated tree in Initial Population.
4. End For
5. For generation g = 1 to G step g+1 do
6. For individual i = 1 to pop step i+1 do
7. For Experiment j=1 to E step j+1 do
8. Generate the initial cloning template ( $p_{ij}$ ) randomly
9. put the learning epochs $L_j = 0$ and the relative fitness $D(p_{ij}) = 0.001$
10. Check the transient, i.e. the computation of the CNN output is stopped if the CNN reaches an equilibrium point or after prescribed number of iterations and Calculate $g_1(p_{ij})$ by using Eq.(7)
11. If ( $g_1(p_{ij}) \neq 0$ ) then
12. $L_j = L_j + 1$
```

```
13. Modify the template $p_{ij}$ according to the tree supervised rules, the program tree $i$
14. If ( $L_j \le N_{\max}$ ) GOTO step 10.
15. Else ( $g_1(p_{ij})=0$ )
16. Calculate the relative robustness degree against deviations $D(p_{ij})$
17. End if
18. Put $N_{ij}=L_j$
19. End For
20. Calculate $f_i$ using Eq. (9)
21. Calculate $D(p_i)=\min_{j\in\{1,2,\ldots,m\}}\{D(p_{ij})\}$, then calculate $g_2(p_i)$ by Eq. (11)
22. Calculate $N_i=\max_{j\in\{1,2,\ldots,m\}}\{N_{ij}\}$, and then calculate $g_3(p_i)$ by Eq. (12)
23. End For
24. If (program tree that satisfies Eq. (13) ) Then
25. The program tree is admissible in the current generation.
26. End if
27. Reproduce best set of individual trees based on its fitness and Insert them in Next Generation.
28. While number of new generation population <=pop
29. Use the tournament selection, which choose 10 programs randomly and return the two better individuals among of them, then randomly choose one of the three trees and its corresponding in the next individual and create one new individual from the two existing individuals by genetically crossover operation.
30. Create new individual from an existing  by randomly mutating a randomly chosen part of the selected tree using mutation operator, replace the internal node by another one i.e. if it was + for example it can be replaced by  - or *.
31. Insert the tree into the in Next Generation.
32. End while
33. Output the best so far individual (The individual that has the highest fitness value overall the generations and minimize the second fitness value in Eq. (13) ) is designed as the result of the run.
34. End For
END
```

Table : GP in learning the CNN new learning rules

Objective	Find a program trees that can be used to discover 3 supervised learning rules corresponding to the feed forward, feedback and the threshold templates.
Architecture of the overall program	Three different trees are chosen corresponding to the feed-forward, the feed-back, and the threshold respectively.
Terminal Set (τ_1) for the first tree	It is considered in accordance with the feed-forward cloning template. It is considered to be a function in the input U that belong to the sphere of influence $N_r(ij)$, for our case we choose r=1, and the parameters, the error and the first order derivative of the activation function
Terminal Set (τ_2) for the second tree	It is considered in accordance with the feed-back cloning template. It is considered to be a function in the output that belongs to the sphere of influence and the parameters, the error and the first order derivative of the activation function.
Terminal Set (τ_3) for the third tree	The last tree is considered in accordance with the threshold. It is considered to be a function in the parameters only, the error and the first order derivative of the activation function, where it depends only on the cell itself.
Function Set without ADF(F)	+,-, and *.
Raw fitness	The raw fitness is considered as the cost function $G(p_1,p_2,\ldots,p_m)=\sum_{i=1}^{m} w_i g(p_i),\ \sum_{i=1}^{m} w_i=1$, $g(p)=\sum_{i=1}^{S_i}(y_i^d-y_i(\infty))^2 \quad w_i\in(-1,1)$ and $S_i=MN$ is the CNN size. Coupled to this, it should satisfy the hardware implementation to be maximized $D(p_i)=\min_{j\in\{1,\ldots,m\}}\max_{\alpha}\{\alpha \mid y_\infty(p_{ij}\circ(1+\alpha 1^{\pm})) = y_\infty(p_{ij}) \text{ for all } 1^{\pm}\in\beta^j\}$ in addition to have the minimum number of learning epochs $N_i=\max_{j\in\{1,2,\ldots,m\}}\{N_{ij}\}$

Standardized Fitness	The cost function, which is minimized, is mapped into fitness value $f_i(.)$ to be maximized by GP. The fitness of the tree i is $f_i = K - G(p_1, p_2, ..., p_m)$, $K = 4\sum_{i=1}^{m} w_i S_i$ since the quadratic value of the error is bounded by $0 \le (y_i^d - y_i(\infty))^2 \le 4$. Coupled to this we should minimize $\lambda g_2(p_i) + (1-\lambda) g_3(p_i)$ for all value of i , where $g_2(p_i) = \begin{cases} (1-\log_{10}(D')) & 0.1\% \le D \le 10\% \\ 0 & D > 10\% \end{cases}$, $D' = 100D(p_i)$ and $g_3(p_i) = \begin{cases} 1-\exp(-\frac{abs(N_i - N_{\min})}{N_{\min}}) & if \;\; N_i \le N_{\min} \\ 1-\exp(-(\frac{abs(N_i - N_{\min})}{N_{\max - N_{\min}}})^2) & if \;\; N_i > N_{\min} \end{cases}$

4. SIMULATION RESULTS

The supervised learning rules program has been developed as C# code [19]. GP evaluates every program by calculating the transient of the CNN defined by the templates that are created from the program trees. If the cost function converges to the tolerance then the process is stopped learning, otherwise CNN template (A, B, z) is enhanced based on the program tree however the CNN transient is checked again. The process is repeated until a prescribed number of learning epochs is counted. Depending on the example the GP parameters are chosen and GP operators to be evolved for each generation. Experiments were conducted under the following conditions:

A. The maximum number of learning epochs is $N_{\max} = 500$, the minimum number of learning epochs is $N_{\min} = 50$, the learning rate is chosen as a random number between 0 and 0.5, the maximum depth for creating trees= 6, maximum depth of crossover operator=17, crossover rate= 90% , mutation rate 1%, the population size $pop = 1000$ and the generation size=200.

B. A fully connected Space invariant CNN with linear synaptic is used.

C. The experiment was conducted on 2 different experiments, $m = 2$, the average scaling CNN and the CNN convex corner detection;

1) The maximum number of learning epochs is $N_{\max} = 500$, the minimum number of learning epochs is $N_{\min} = 50$, the learning rate is chosen as a random number between 0 and 0.5, the maximum depth for creating trees= 6, maximum depth of crossover operator=17, crossover rate= 90% , mutation rate 1%, the population size $pop = 1000$ and the generation size=200.

2) A fully connected Space invariant CNN with linear synaptic is used.

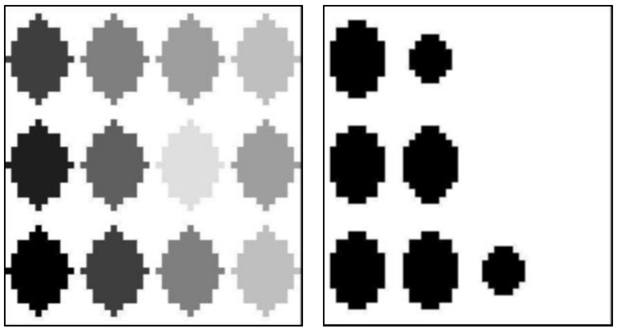
3) The experiment was conducted on 2 different experiments, $m = 2$, the average scaling CNN and the CNN convex corner detection;

a. $u_{ij} = 1$ maps into $y_{ij} = 1$ if, and only if , there are exist, at least 5 nearest neighbors with $u_{ij} = -1$.

b. $u_{ij} = -1$ maps into $y_{ij} = -1$ independent of its neighbors.

c. The task was an 24×24 convex corner detection with fixed boundary condition and initial state equal zero $x_{ij}(0)$ =0.

d. To learn the task, the feed-forward and feed-back cloning template of the CNN were first initialized to random values (to be sure our learning rule is able to solve the task from many initial conditions).



(a) The input image *(b) The desired output*

Fig. 7. The average Scaling CNN training Data

As the result of running our experiment we get the admissible result in the fourth trail on generation number 21, with error free and number of learning epochs 328 and relative fitness greater than 0.1, with the following trees;

- (*(+(+(+(DRV Ykl) -(*(+(DRV -(DRV *(Ykl ERR))) +(-(+(DRV Ykl) Ykl) ERR)) Ykl)) +(+(+(-(-(+(DRV Ykl) Ykl) -(+(ERR DRV) DRV)) -(*(*(ERR ERR) *(ERR ERR)) DRV)) ERR) -(*(DRV +(ERR +(ERR Ykl))) *(+(-(-(-(DRV DRV) Ykl) *(ERR ERR)) -(*(*(ERR ERR) *(ERR ERR)) DRV)) Ykl)))) DRV)).
- (*(+(+(Ukl *(*(*(*(+(+(Ukl *(-(*(*(*(ERR Ukl) *(ERR Ukl)) DRV) -(Ukl *(*(DRV

ERR) Ukl))) Ukl)) Ukl) ERR) DRV) -(Ukl *(
*(DRV ERR) Ukl))) Ukl)) Ukl) ERR))
• (+(+(ERR DRV) –(DRV DRV))).

These three trees can be summarized as follows;

$$A(kl) = A(kl) + \eta \sum_{i=1}^{M} \sum_{j=1}^{N} \begin{pmatrix} (f`(x_{ij})^2 + 2f`(x_{ij})^3 + 4f`(x_{ij})^2 Error + f`(x_{ij}) Error^4) + \\ (2f`(x_{ij})^2 - f`(x_{ij})^2 Error - f`(x_{ij}) Error^4) y_{kl} + f`(x_{ij}) y_{kl}^2 \end{pmatrix}$$

(16)

$$B(kl) = B(kl) + \eta \sum_{i=1}^{M} \sum_{j=1}^{N} \begin{pmatrix} 2Error^* u_{kl} + (2f`(x_{ij}) Error - 2f`(x_{ij})^2 Error^3) u_{kl}^3 + \\ (-f`(x_{ij}) Error^2 + 2f`(x_{ij})^2 Error^3 - f`(x_{ij})^3 Error^4) u_{kl}^4 + \\ (f`(x_{ij})^2 Error^4 - f`(x_{ij})^3 Error^5) u_{kl}^5 \end{pmatrix}$$

(17)

$$z = z + \eta \sum_{i=1}^{M} \sum_{j=1}^{N} \left(f`(x_{ij}) + Error \right) \quad (18)$$

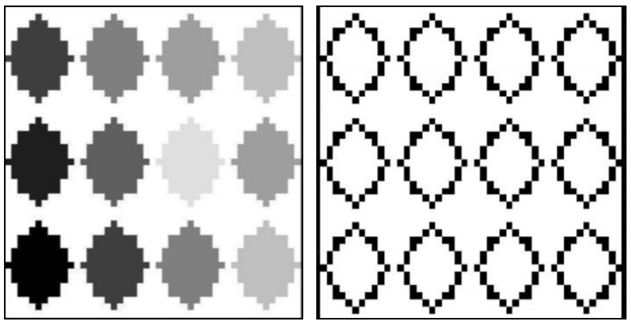
where η is the appropriate learning rules, which is a constant for all the 3 learning rules, $\eta \in (0,1)$, and $Error = (y_{ij}^d - y_{ij}(\infty))$.

4.1 Example 1 (Edge Gray CNN problem)

To be able to discover the ability of new supervised learning rules, a complex and famous application should be solved. Gray scale input image is selected. It comprises too much redundancy and needs many more "bits" to be represented than binary image. Moreover, the processed output of gray image may not yield binary image. Edge-gray CNN problem receives gray scale input image and processed for Edge detection, binary output image. The Edge-gray problem is illustrated in Fig. 7 where (a) and (b) state the gray-scale input and binary output images respectively. For any gray-scale input Image *U*, assuming $x_{ij}(0) = 0$, the black pixels are assumed in accordance with pixels laying on sharp edges of *U*. Also, the black pixels are laying in vague edges that are combination of gray pixels of *U* such that the intensity of pixels on one side of the arc differs significantly from the intensity of neighbor pixels of the other side of the arc.

The edge gray CNN can be recognized by uncoupled CNN. Therefore, the feedback template is considered to be zero except the self-feedback. By taking the learning rate to be $\eta = 0.02$, we discover the templates with zero error after 150 of learning epochs to be;

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3.4339 & 0 \\ 0 & 0 & 0 \end{pmatrix}, B = \begin{pmatrix} -0.1790 & -2.1939 & -0.1966 \\ -2.5127 & 9.7488 & -2.0182 \\ -0.2259 & -2.2263 & -0.2464 \end{pmatrix} \text{and } z = -0.3445$$



(a) The input image *(b) The desired output*

Fig. 7. The Edge-gray CNN problem

In contrast with The Truncation Learning of CNN, experiment under the same conditions is performed for both techniques, learning rate and input pattern. As depicted in Fig. 8, where NSLCNN refers to the New Supervised Learning and TL refers to the Truncation Learning, we found that the new method has faster convergence than that of Truncation learning. Also as illustrated in Fig. 9, by increasing the number of epochs, the Truncation Learning cumulative error is monotonically increasing function on iteration while the error discovered by new learning rules remains near to zero error. Hence the new supervised learning rules NLSCNN can discover more robust template than that of TL. Moreover, it is faster to discover the optimal templates.

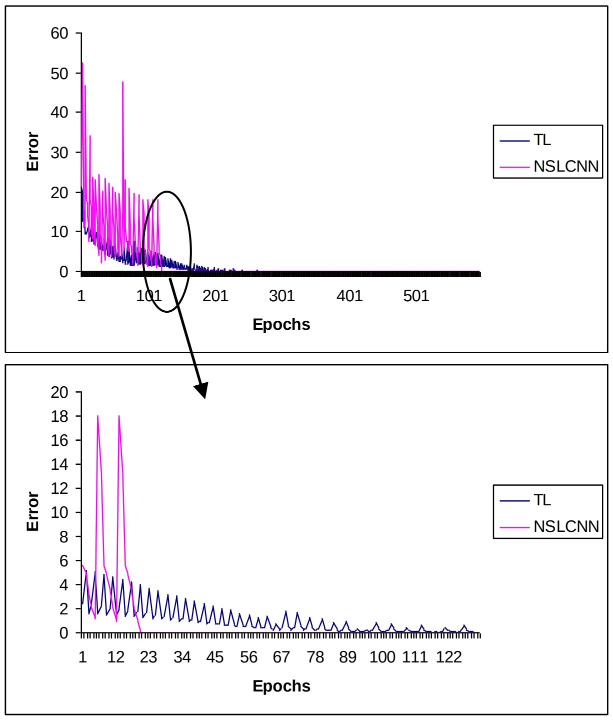


Fig. 8. The convergence speed diagram.

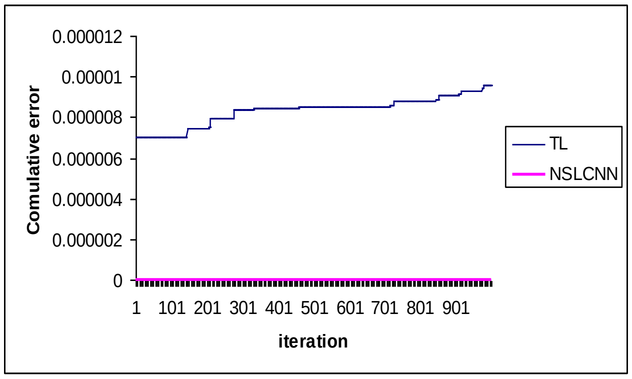
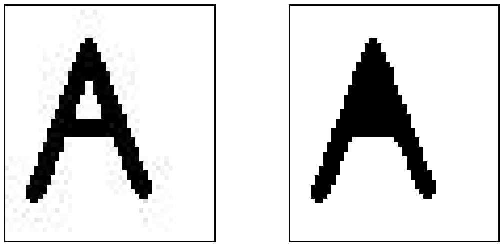


Fig. 9. The error diagram

4.2 Example 2 (Auto fill template)

Our next example is the hole filler CNN. A given bipolar image, representing a black object on a white background, is fed into the input while all initial states are equal to 0. Then the CNN must fill up any hole in the object, as demonstrated in Fig. 10 where (a) and (b) are the input pattern and the desired output pattern respectively.



(a) The input image *(b) The desired output*

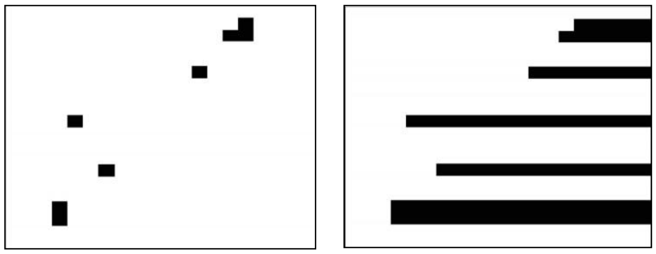
Fig. 10. Hole filling of a black object on a white background

As a result of using the NSLCNN rules by learning rate equal to 0.06, we get the template with error $2.28*10^{-15}$ after 168 of learning epochs by the following form.

$$A=\begin{pmatrix}3.609 & 6.241 & 2.206\\ 6.162 & 3.737 & 3.208\\ 4.251 & 6.205 & 4.723\end{pmatrix}, B=\begin{pmatrix}-0.872 & -0.556 & -0.179\\ -1.726 & 11.926 & -0.519\\ -0.0896 & -0.924 & -0.397\end{pmatrix} \text{and } z=6.165$$

4.3 Example 3 (Shadowing)

The shadow detector [8] is presented in this example. As shown in Fig. 11 where (a) and (b) represent input image and the desired output, all the pixels located right in each row for black pixels should become black. In the training set the initial state is chosen to be the input pattern.



(a) The input mage *(b) The desired output*

Fig. 10. Hole filling of a black object on a white background

The task was learned with 20×20 training example and the learning rate $\eta = 0.14$. As a result of using the NSLCNN rules, we get the template with error $1.262*10^{-23}$ after 361 of learning epochs by the following form.

$$A=\begin{pmatrix}0 & 0 & 0\\ 16.58 & 21.06 & 0\\ 0 & 0 & 0\end{pmatrix}, B=0 \text{ and } z=29.070$$

5. CONCLUSIONS AND FUTURE WORK

In this paper, Genetic Programming (GP) has been applied to find new supervised learning rules for Cellular Neural Networks. GP program is defined to be three different trees that are evolved in accordance to feed forward, feedback and search bias supervised learning rules. The optimization problem has assumed multi-objective optimization problem by taking into account the hardware implementation as a constraint in the optimization problem. Besides, the GP raw fitness for each program tree has been considered as linear combination among different experiments to handle the whole cases of the CNN templates. GP has discovered new supervised learning rules that have performed much better than Truncation Learning, mathematical derivation supervised rules, on all sample problems. The new admissible learning rules are applied into three different experiments, the edge gray CNN, the hole filling and shadowing detection, in purpose of proving the ability of the new supervised rules.

Although these rules have achieved a great progress in learning the template for CNN, they suffer from different problems, such as their learned template values that depend on the training image size. Coupled to this, they can't discover the optimal templates structure by themselves. This gives us a new direction of research to combine GP with the machine learning algorithms that are able to reduce the learning values by discovering the optimal template structure by removing the superfluous cells and indicating the similarities and regularities in data. With these new rules, the performance of CNNs is enhanced. Simulation results have confirmed the theoretical considerations. Moreover, comparisons with related work have been given. A hybrid model of

Reinforcement optimization technique based on GP will be used in complex system such as SCADA systems.

REFERENCES

[1] A. Adamatzky, “Reaction-Diffusion Automata: Phenomenology, Localisations, Computation, Springer Publishing Company”, Incorporated, 2012.

[2] L. O. Chua, and L. Yang, “Cellular Neural Networks: applications”, IEEE Transaction on Circuits and System. Vol. 35, No. 10, Oct., 1988, pp. 1257-1272.

[3] K. Balasubramaniam, B. Luitel and G. K. Venayagamoorthy, “A Scalable Wide Area Monitoring System using Cellular Neural Networks”, WCCI 2012 IEEE World Congress on Computational Intelligence, Brisbane, Australia, June, 10-15, 2012.

[4] A. A. ALbahbah, H. M. El Bakry, and S. Abd-Elgahany, “Detection of Caries in Panoramic Dental X-ray Images using Back-Propagation Neural Networks", International Journal of Electronics Communication and Computer Engineering, vol. 7, issue 5, September 2016, pp. 250-256.

[5] G. Liang, X. Wu and W. Du, “New Condition for Global Asymptotically Stability of Cellular Neural Networks with Time Delay” 2010 Sixth International Conference on Natural Computation (ICNC 2010), 2010, pp. 785-789.

[6] Bartfai, “Fault-Tolerant Design of Analogic CNN Templates and Algorithms- Part I: The Binary Output Case”, IEEE Transactions on Circuit and Systems- I: Fundamental Theory and Applications, vol. 46, no. 2, 1999, pp. 312-322.

[7] V.P. MASLOVE, W. D. NEUMANN, R.O. WELLS, “STABILITY ANALYSIS OF IMPULSIVE FUNCTIONAL DIFFERENTIAL EQUATIONS”, WALTER DE GRUYTER GMBH, BERLIN, GERMANY, 2009.

[8] T. Matsumoto, L. O. Chua, and H. Suzuki, “CNN cloning template: shadow detector”, Transaction on Circuits and Systems, vol. 37, 1990, pp. 1070-1073.

[9] L. Fortuna, M. Frasca and M.G. Xibilia, “Chua’s circuit implementations: Yesterday, today and tomorrow”, World Scientific series on nonlinear science. Series A. ; v. 65, 2009.

[10] L. O. Chua, and T. Roska, “Cellular Neural Networks and Visual Computing”, Cambridge University Press, 2002.

[11] E. Radwan, M. Tarek, A. Baz, “Distributed Optimization Model of Wavelet Neuron for Human Iris Verification”, International Journal of Advanced Computer Science and Applications, Vol. 6, No. 12, 2015, pp. 209-218.

[12] J. R. Koza, “Genetic Programming, On the Programming of Computers by Means of Natural Selection”, The MIT Press, 1992.

[13] J. Branke, S. Nguyen, C. W. Pickardt and M. Zhang, "Automated Design of Production Scheduling Heuristics: A Review", IEEE Transactions on Evolutionary Computation, vol. 20, no. 1, Feb. 2016, pp. 110-124.

[14] R. Riolo , E. Vladislavleva, M. D. Ritchie and J. H. Moore, “ Genetic Programming Theory and Practice XIII”, Springer Science Business Media, New York, 2016.

[15] L. O. Chua, “CNN: A vision of complexity”, International Journal of Bifurcation and Chaos, vol. 7, no. 10, 1997, pp. 2219-2425.

[16] B. Luitel, G. K. Venayagamoorthy, "Decentralized asynchronous learning in cellular neural networks", IEEE Trans. Neural Netw. Learn. Syst., vol. 23, no. 11, 2012, pp. 1755-1766.

[17] V. M. Preciado, D. Guinea, , J. Vicente, M. C. Garcia-Alegre and A. Ribeiro , “Genetic Programming of a CNN Multi-Template Tree for Automatic Generation of Analogic Algorithms”, Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications Catania, 2000, pp. 381-386.

[18] P. J. Angeline, K. E. Kinnear, and Jr., “Advances in Genetic Programming 2”, The MIT Press, Ch. 13, 1996.

[19] E. Radwan and E. Tazaki, “Template Learning of Cellular Neural Networks Using Genetic Programming”, International Journal of Neural Systems, vol. 14, no. 4, 2004, pp. 247-256.

[20] E. Radwan, and E. Tazaki, “New learning method for Cellular Neural Networks based on combination between Rough Sets and Genetic Programming”, Cybernetics and Systems, International Journal CBS, vol. 36, no.4, 2005, pp. 415-444.

[21] W. B. Langdon, “Genetic Programming and Data Structure”, Kluwer Academic Publishers, 1998.

[22] I. El-Henawy, H. M. El-Bakry, H. M. El Hadad, and N. Mastorakis "Muzzle Feature Extraction Based on gray level co-occurrence matrix," International Journal of Veterinary Medicine, vol.1, December 2016, pp.16-24.

[23] P. Lopez, D.L. Vilarino, V.M. Brea and D. Cabello, “Robustness oriented design tool for Multi-layer DTCNN applications”, International Journal of Circuit Theory and Applications, vol. 30, 2002, pp. 195-210.

[24] A. Paasio and A. Dawidziuk, “CNN template robustness with different output nonlinearities”, International Journal of Circuit Theory and Applications, vol. 27, 1999, pp. 87-102.