2009 IEEE Workshop on
Nonlinear Circuit Networks, Tokushima, Dec 11-12, 2009

# CNN Template Learning To Obtain Desired Images Using Back Propagation Algorithm

Masashi Nakagawa, Takashi Inoue and Yoshifumi Nishio
Department of Electrical and Electronic Engineering, Tokushima University
2-1 Minami-Josanjima, Tokushima, 770-8506, Japan
Email: {miyabi, takashi, nishio}@ee.tokushima-u.ac.jp

## I. Introduction

Cellular neural networks (CNN) were introduced by Chua and Yang in 1998 [1]. The idea of the CNN was inspired from the architecture of the cellular automata and the neural networks. Unlike the conventional neural networks, the CNN has local connectivity property. Since the structure of the CNN resembles the structure of animals retina, the CNN can be used for various image processing applications [2]-[4].

Wiring weights of the cells of the CNN are established by parameters called the template. The template is most important parameter, because performance of the CNN is decided by the template. Thus, some template design methods as template learning using GA algorithm are proposed. These works are important subject in the studies of the CNN [5].

In this paper, template learning of cellular neural networks using back propagation algorithm is proposed. In our proposed CNN, we build back propagation algorithm into the feedback part of CNN. Back propagation algorithm is inspired form back propagation neural networks [6]. Back propagation neural networks operates with a feed forward neural network which is composed of an input layer, a hidden layer and an output layer, and the effectiveness of the back propagation algorithm has been confirmed in learning performance. In this paper, the template of CNN is dynamically updated in each step using the back propagation algorithm. Computer simulations using one-dimensional image show that the back propagation algorithm is effective for template learning of CNN. At the moment, we do not say that the proposed template learning method exhibited a superior performance than the other template learning method. However, we feel that we obtained some results to broaden the research on the template learning of CNN.

## II. Proposed Template Learning Method

In this section, we describe the proposed template learning method using back propagation algorithm. Our template learning method is inspired from perception of the back propagation algorithm. In particular, in the back propagation algorithm, parameter is changed by using error between an output and a desired value, in order to bring an output close to a desired value. From this perception of learning, template of CNN is changed by using error between an output image and a desired image. Additionally, this template learning carry out dynamically. Namely, processing by the CNN and template learning are carried out at the same time.

### A. Processing Procedure for Learning

Processing procedure for learning is explained as follows.

Step 1. An input image and a desired image (a teacher image) are prepared. An initial template is decided at random. Because we use random in order to be able to learn the template for any initial values. However, in our learning method, we should mention the stability of template in CNN. Because the value of an output may become unstable with the value of a template. The condition of stability is that the center element of template $A$ maintains 1 and the symmetry of each element.

Step 2. An input image is processed by CNN using Runge-Kutta method. At this time, unit width of Runge-Kutta method is made small, and only a few processes. It is for learning dynamically not using the processed image but using the present image currently processed. At this time, the value of $Vx$ before processing is defined as $Vx^{past}$, and the value of $Vx$ after processing is defined as $Vx^{now}$. It is for deciding the direction of correction by comparing with the value of $Vx$.

Step 3. The error between an output image and a desired image is calculated for every pixel of an output and a desired image. An absolute value is calculated in order to make the error of each pixel into a positive value.
An error $E_p$ is calculated as the following equation:

$$E_p = \sqrt{(t_p - o_p)^2}, \tag{1}$$

where $p$ is the number of the pixel, $t_p$ denotes the value of the desired data of the $p$th pixel, and $o_p$ denotes the value of the output data of the $p$th pixel.

Step 4. The corrected direction $\delta_p$ is decided using error $E_p$ as the following equation:

$$\delta_p = E_p \times (Vx_p^{now} - Vx_p^{past}) \tag{2}$$

Step 5. The correction quantity $\Delta_p$ for every pixel is calculated as the following equation:

$$\Delta_p = \eta \times \delta_p \times Vx_p^{now}, \tag{3}$$

where $\eta$ is a proportionality factor known as the learning rate.

Step 6. The errors of all the pixels are added.

$$D = \sum_{i=1}^{p} \Delta_i \tag{4}$$

The calculation adds all the values of a template. However, it does not include the center element of the template $A$, $B$, and the bias $I$. That is, a template is changed as the following equation.

$$\begin{aligned} A_{10}^{Renewed} &= A_{10}^{Before} + D, \\ A_{12}^{Renewed} &= A_{12}^{Before} + D, \\ B_{10}^{Renewed} &= B_{10}^{Before} + D, \\ B_{12}^{Renewed} &= B_{12}^{Before} + D \end{aligned} \tag{5}$$

The reason of the above setting is that the stability of the template can be kept.

Step 7. Step.2 - Step.6 are repeated until the process converges.

Thus, by updating a template, an output image is brought close to a desired image.

Finally, the diagram showing the flow of the process is shown in Fig. 1.

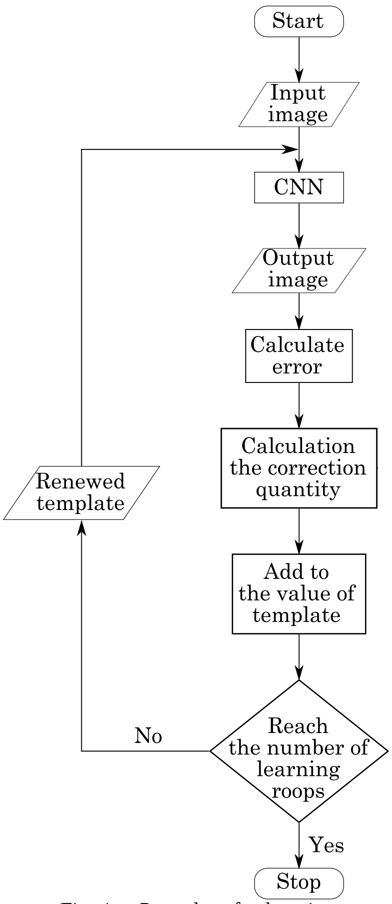


Fig. 1. Procedure for learning

## III. Simulation

In this section, we show the learning results using the proposed learning method.

### A. Input Image and Desired Image

We prepare a one-dimensional input image and desired image as Figs. 2 and 3.

*Input image:*

Fig. 2. Input image

*Desired image:*

Fig. 3. Desired image

The black pixels correspond to the value +1, while the white pixels correspond to the value -1.

A teacher image is made from the following two rules.

Rule 1. If the colors of the neighbors of a certain cell is the same color, the cell's color turns into its neighboring color.

Rule 2. If the colors of the neighbors of a certain cell is different, the cell's color does not change.

That is to say, this rule is like "Reversi".

### B. Parameters for Learning

The following parameters are used in this simulation.

TABLE I
PARAMETERS

| Name of parameter | value |
|---|---|
| Step size | 0.01 |
| Maximum iteration number | 10 |
| Number of pixels | 30 |
| Number of learning loops | 100 |
| Learning rate | 0.01 |

### C. Simulation Results

As a result of carrying out computer simulations, the following images and templates are obtained. In our proposed learning method, because elements of the initial template are set at random. Therefore, the results are always different. The results shown below are some examples of obtained results.

*Learning result 1:*

Figure 4 shows the example of the output image using the proposed learning method.

Fig. 4. Result image

In Fig. 4, we can see that the three pixels differ from the desired image. In this case, we obtained the template as follows.

*Obtained template:*

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0.917804 & 1 & 0.917804 \\ 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 1.538764 & 0.113712 & 0.644816 \\ 0 & 0 & 0 \end{bmatrix},$$

$$I = 0 \tag{6}$$

In obtained template (6), elements of feedback template $A$ stability.

*Learning result 2:*

The following results are obtained by using the same parameters as the learning result 1.

Fig. 5. Result image

In Fig. 5, the same result as the desired image can obtained. The different template outputting the desired image are obtained by the proposed learning method.

*Obtained template:*

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0.712821 & 1 & 0.712821 \\ 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 1.425642 & 0.7734 & 1.911821 \\ 0 & 0 & 0 \end{bmatrix},$$

$$I = 0 \tag{7}$$

Although the learning parameters are the same, different templates are obtained. Because the initial values of the templates were decided at random.

The “Reversi” was able to be performed when processing other input images by the conventional CNN using the obtained templates as template (7). However, since these templates are updated dynamically, it is not a template in the conventional CNN which can certainly realize “Reversi”. Additionally, we should mention that we obtained the template performing “Reversi” only for working one-dimensional images in this study.

## IV. Conclusions

In this paper, we have proposed a template learning method of cellular neural networks (CNN) using the back propagation algorithm. The template was learned by using the error between a desired image and an output image. Moreover, it was able to bring close to the desired image, changing a template by processing dynamically using back propagation algorithm. However, since this proposed method has the characteristic of processing dynamically, a desired image is not necessarily obtained. Therefore, we want to obtain a desired image by raising accuracy.

In the future, we want to learn the template which can carry out processing two-dimensional images, and processing gray scale images using back propagation algorithm.

## References


[1] L.O. Chua and L. Yang, “Cellular Neural Networks:Theory,” IEEE Trans. Circuits Syst., vol. 32, pp. 1257-1272, Oct. 1988.

[2] F. Dirk and T. Ronald, “Coding of Binary Image Data using Cellular Neural Networks and Iterative Annealing,” Prof. of ECCLES, 03, vol. 1, pp. 229-232, Se. 2003.

[3] M. Samba and Z. Hang, “Cellular Neural Network for Associative Memory and Its Application to Braille Image Recognition,” Prof. of 06, pp. 4716-4721, Jul. 2006.

[4] H. Koeppl and L.O. Chua, “An Adaptive Cellular Nonlinear Network and its Application,” Proc. of NOLTA’ 07, pp. 15-18, Sep. 2007.

[5] T. Kozek, T. Roska, and L.O. Chua, “Genetic Algorithm for CNN Template Learning,” IEEE Trans. Circuits Syst., vol.40, pp. 392-401, Jun. 1993.

[6] D.E. Rumelhart, G.E. Hinton and R.J. Williams, “Learning Internal Representations by Error Propagation,” Parallel Distributed Processing, vol. 1, MIT Press, MA, pp. 318-362, 1986.