2011 International Workshop on Nonlinear Circuits,
Communication and Signal Processing
NCSP'11, Tianjin, China, March 1-3, 2011

Modified Back Propagation Algorithm for CNN Template Design

Masashi Nakagawa[†], Yoko Uwate[†] and Yoshifumi Nishio[‡]

†Tokushima University
2-1 Minami-Josanjima, Tokushima, Japan
Phone:+81-88-656-7470, Fax:+81-88-656-7471
Email: {miyabi, uwate, nishio}@ee.tokushima-u.ac.jp

Abstract

In the previous study, we have proposed a template design method of cellular neural networks with back propagation algorithm. In that method, template learns by using the average error which corresponds to the difference between the output image and the desired image. In this study, we modify the back propagation algorithm for cellular neural networks template design. We inspect the performance of this learning algorithm for two-dimensional binary image.

1. Introduction

Cellular neural networks (CNN) were introduced by Chua and Yang in 1998 [1]. The idea of the CNN is 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.

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 determined by the template. Thus, some template design methods, e.g., template learning using genetic algorithm (GA), have been proposed. These works are important subject in the studies of the CNN.

In the previous study, we have proposed a template learning method of CNN using the back propagation (BP) algorithm [2]. As well known, BP is one of the supervised learning methods [3]. In that method, the template of CNN is dynamically updated from the error between an output image and a desired image as BP neural networks. We have confirmed that the method could work for a simple example using one-dimensional binary image by computer simulations. Furthermore, we have investigated the method for gray scale image. However, in the case of applying for gray scale images, we have not obtained good results. This is because the previous method was a technique of imitating the process of conventional BP. Therefore, even if the method was able to process binary images, the method was not able to process gray scale images. Thereupon, in this study, we modify the back propagation algorithm for CNN template design. In addition, we examine the performance of this learning algorithm for two-dimensional binary image.

2. Cellular Neural Networks

In this section, we explain the basic structure of the CNN. The CNN has M by N processing unit circuits called cells. The cells are arranged in a reticular pattern to M line N row. We represent a cell $C(i, j)$ using a variable i which denotes vertical position and a variable j which denotes horizontal position. The cell contains linear and nonlinear circuit elements. The CNN is an array of cells. Each cell is connected to its neighboring cells according to a template. Usually, the template is the same for all cells except for boundary cells. The CNN has the features of time continuity, spatial discreteness, nonlinearity and parallel processing capability.

The state equation and the output equation of the cell are shown as the following equations.

State equation:

$$\begin{aligned}\frac{dv_{xij}(t)}{dt} &= -v_{xij}(t) + \sum_{k=i-r}^{i+r}\sum_{k=j-r}^{j+r} A_{(i,j;k,l)}v_{ykl}(t) \\ &\quad + \sum_{k=i-r}^{i+r}\sum_{k=j-r}^{j+r} B_{(i,j;k,l)}v_{ukl}(t) + I \qquad (1)\end{aligned}$$

Output equation:

$$v_{yij}(t) = \frac{1}{2}(|v_{xij}+1| - |v_{xij}-1|) \qquad (2)$$

where v_x, v_y and v_u represent a state, an output and an input of cell, respectively. In the equation (1), A is the feedback template and B is the control template. These and bias I are collectively called general template.

3. Proposed Template Learning Algorithm

In this section, we explain the proposed template learning algorithm based on BP algorithm. Our template learning algorithm is considered based on the dynamic and flexible point in BP algorithm. In particular, in BP neural networks, parameter is changed by using error between an output signal and a desired signal, in order to bring an output close to a desired value. From this learning algorithm, the template of CNN is changed by using the error between an output image and a desired image, and we can obtain a desired image. Moreover, since the template in this study is always continuing learning, we can refer that the template is dynamical.

In the previous study, we have proposed some learning methods [4]. However, these learning methods have imitated the process of the original BP algorithm. Because of this, better results have not been obtained in one-dimensional gray scale images.

3.1. BP Algorithm

In the neural networks, the weight is adjusted according to the following equation.

$$\Delta_p w_{ji} \propto -\frac{\partial E_p}{\partial w_{ji}}. \tag{3}$$

Equation (3) is taken apart as follows,

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial x_{pj}} \frac{\partial x_{pj}}{\partial w_{ji}}, \tag{4}$$

$$(x_{pj} = \sum_i w_{ji} o_{pi}). \tag{5}$$

The second differential calculus of the right side of Eq. (4) is calculated as follows,

$$\frac{\partial x_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_k w_{jk} o_{pk} = o_{pi}. \tag{6}$$

We define an error direction δ_{pj} as follows,

$$\delta_{pj} = -\frac{\partial E_p}{\partial x_{pj}}. \tag{7}$$

In conclusion, we adjust the weight according to the following equation,

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}. \tag{8}$$

Next, we explain how to decide the rate of change δ_{pj}.In order to calculate Eq. (7), we divide as follows,

$$\delta_{pj} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial x_{pj}}. \tag{9}$$

Then, in the neural networks, o_{pj} is expressed as follows,

$$o_{pj} = f_j(x_{pj}), \tag{10}$$

$$(f_{(x)} = \frac{1}{1+e^{-\beta x}}), \tag{11}$$

the second differential calculus of the right side of Eq. (9) is calculated as follows,

$$\frac{\partial o_{pj}}{\partial x_{pj}} = f'_j(x_{pj}). \tag{12}$$

Next, the first differential calculus of the left side of Eq. (9) is calculated as follows,

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}). \tag{13}$$

In conclude of all, δ_{pj} is expressed as follows,

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(x_{pj}). \tag{14}$$

In this way, in the neural networks, the error is able to calculate.

When we use sigmoidal function as output function in neural networks, the rate of change δ_{pj} is expressed as:

$$\delta_{pj} = \beta(t_{pj} - o_{pj}) o_{pj} (1 - o_{pj}). \tag{15}$$

3.2. Approximation of Output Equation

First, we approximate the output equation of CNN. Output equation of CNN is shown as follows.

$$v_{y_p} = \frac{1}{2}(|v_{x_p} + 1| - |v_{x_p} - 1|) \tag{16}$$

Since the output equation of CNN is piece-wise linear function, we cannot differentiate it. So, we divide output equation to 3 parts as follows.

Part 1 : When $v_x \le -1$, $v_y = -1$
Part 2 : When $1 \le v_x$, $v_y = 1$
Part 3 : When $-1 \le v_x \le 1$, $v_y = v_x$

Especially, we pay attention to [Part 3]. Because, in this part, the values of v_x and v_y change according to $y = x$. We consider that this part is more important for learning template than the other parts.

3.3. Modified BP Algorithm for Template Learning

First, we consider templates A and B as variables as follows,

$$\Delta A_{ij} = -\frac{\partial E_{ij}}{\partial A_{ij}} = -\frac{\partial E_{ij}}{\partial v_{x_i j}} \frac{\partial v_{x_i j}}{\partial A_{ij}}. \tag{17}$$

The first differential calculus of Eq. (17) is calculated as follows,

$$\delta_{ij} = -\frac{\partial E_{ij}}{\partial v_{x_i j}} = -\frac{\partial E_{ij}}{\partial v_{y_i j}} \frac{\partial v_{y_i j}}{\partial v_{x_i j}}. \tag{18}$$

The each differential calculus of Eq. (18) is calculated as follows,

$$\frac{\partial E_{ij}}{\partial v_{y_{ij}}} = -(t_{ij} - o_{ij}), \tag{19}$$

$$\frac{\partial v_{y_{ij}}}{\partial v_{x_{ij}}} = 1. \tag{20}$$

In conclude of all,

$$\delta_{ij} = -\frac{\partial E_{ij}}{\partial v_{x_{ij}}} = (t_{ij} - o_{ij}) \times 1. \tag{21}$$

Next, the second differential calculus of Eq. (17) is calculated as follows. To begin with, we consider the state equation of CNN,

$$\frac{\partial v_{x_{ij}}}{\partial t} = -v_{x_{ij}} + \Sigma(Av_{y_{ij}}) + \Sigma(Bv_{u_{ij}}) + I. \quad (22)$$

We exchange an element of Eq. (22),

$$v_{x_{ij}} = -\frac{\partial v_{x_{ij}}}{\partial t} + \Sigma(Av_{y_{ij}}) + \Sigma(Bv_{u_{ij}}) + I. \quad (23)$$

We differentiate Eq. (23) with templates A and B,

$$\frac{\partial v_{x_{ij}}}{\partial A_{ij}} = -\frac{\partial^2 v_{x_{ij}}}{\partial t \times \partial A_{ij}} + v_{y_{ij}}, \quad (24)$$

and we assume that $-\frac{\partial^2 v_{x_{ij}}}{\partial t \times \partial A_{ij}}$ is 0.

In conclude, in the case of template A,

$$\frac{\partial v_{x_{ij}}}{\partial A_{ij}} = v_{y_{ij}}, \quad (25)$$

Similarly for template B,

$$\frac{\partial v_{x_{ij}}}{\partial B_{ij}} = v_{u_{ij}}. \quad (26)$$

In conclude of all,

$$\Delta A_{ij} = -\frac{\partial E_{ij}}{\partial A_{ij}} = \eta \times (t_{ij} - o_{ij}) \times v_{y_{ij}}, \quad (27)$$

$$\Delta B_{ij} = -\frac{\partial E_{ij}}{\partial B_{ij}} = \eta \times (t_{ij} - o_{ij}) \times v_{u_{ij}}. \quad (28)$$

As a result, we can refer that we can express equation of correction using the error in CNN based on BP.

3.4. Updating Elements of Template

In this study, since we use two-dimensional images, the elements of a template are expressed as follows,

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & 1 & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix},$$

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}.$$

Template has a roll which influence neighborhood pixel of a certain pixel. For example, we consider a relation with the top, bottom, left and right of a certain element. a_{12} and b_{12} influence the top pixel of a certain pixel and a_{32} and b_{32} influence the bottom pixel of a certain pixel. a_{21} and b_{21} influence the left pixel of a certain pixel, a_{22} and b_{22} influence a certain pixel itself and a_{23} and b_{23} influence the right pixel of a certain pixel.

When we use a certain two-dimensional image, a_{12} and b_{12} influence a pixel from (0, 1) to (255, 256), a_{21} and b_{21} influence a pixel from (1, 0) to (256, 255), a_{23} and b_{23} influence a pixel from (1, 2) to (256, 257) and a_{32} and b_{32} influence a pixel from (2, 1) to (257, 256). (Then, 0 and 257 are neighborhood pixels.) Namely, when the elements of a template are updated, a_{12} and b_{12} are added the average error from (0, 1) to (255, 256) between the desired image and the output image, a_{21} and b_{21} are added the average error from (1, 0) to (256, 255), a_{23} and b_{23} are added the average error from (1, 2) to (256, 257), and a_{32} and b_{32} are added the average error from (2, 1) to (257, 256), respectively.

In Eq. (27) and Eq. (28), we show the error in each template. According to these, we calculate the error in each element of template A and B,

$$\Delta a_{mn}(t) = \begin{cases} 0 \quad (if \quad m = n = 2) \\ \frac{1}{MN} \sum_{1 \le i \le M, 1 \le j \le N} \Delta A_{i+m-2, j+n-2}(t), \end{cases} \quad (29)$$

$$\Delta b_{mn}(t) = \frac{1}{MN} \sum_{1 \le i \le M, 1 \le j \le N} \Delta B_{i+m-2, j+n-2}(t), \quad (30)$$

$$a_{mn}(t+1) = a_{mn}(t) + \Delta a_{mn}(t), \quad (31)$$

$$b_{mn}(t+1) = b_{mn}(t) + \Delta b_{mn}(t), \quad (32)$$

where $m, n \in \{1,2,3\}$.

In addition, in the case of template A, we have to keep the stability condition. Then, we add the average error of two elements to symmetric part of elements.

$$\frac{1}{2}\begin{pmatrix} \Delta a_{11} + \Delta a_{33} & \Delta a_{12} + \Delta a_{32} & \Delta a_{13} + \Delta a_{31} \\ \Delta a_{21} + \Delta a_{23} & 0 & \Delta a_{21} + \Delta a_{23} \\ \Delta a_{13} + \Delta a_{31} & \Delta a_{12} + \Delta a_{32} & \Delta a_{11} + \Delta a_{33} \end{pmatrix} \quad (33)$$

3.5. Flow of Learning

CNN processes an image using Runge-Kutta method. So, we define learning as process per "Step size" in Runge-Kutta method. Learning flow is as follows.

Step 1 : An image is inputted.
Step 2 : An image is processed by CNN in "Step size" of first time.
Step 3 : After process with "Step size" of first time, being processed image is compared with the desired image.
Step 4 : Calculating the error.
Step 5 : Updating a template.
Step 6 : Being processed image is processed by CNN using updated a template in "Step size" of second time.
Step 7 : Step 2 - Step 6 are repeated until reaching "Maximum iteration number".

This method is able to update a template more dynamically. Namely, we can bring dynamically the being processed image close to the desired image while processing.

4. Simulation Result of Two-Dimensional Binary Image

In this study, we examine performance of proposed method with two-dimensional binary image. In this simulation, we examine noise reduction in CNN processing. First, we prepare an input image with noise, and the desired image corresponding to the input image. We have CNN learns the template which can process noise reduction.

The following parameters are used in this simulation (see Table 1).

Table 1: Parameters.

Name of parameter	value
Step size	0.005
Maximum iteration number	10000
Element of initial template	$-1 \sim 1$
Learning rate η	0.1

"Step size" and "Maximum iteration number" are parameters of Runge-Kutta method. Learning rate is η of equation (27) and (28). Finally, the input image and the desired image are 256×256 pixels, and the initial template is determined at random.

4.1. Template Learning

We prepare the following input image. The image is binary image with noise and consisted of simple figures; triangle, circle and square.

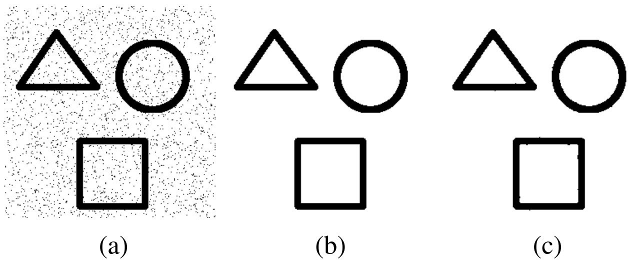
Figure 1: The images of simulation results. (a) Input image. (b) Desired image. (c) Output image.

Figure 1(a) shows the input image with noise. Figure 1(b) shows the desired image. Figure 1(c) shows the output image using the proposed method. Comparing Fig. 1(c) with Fig. 1(b), some noises remain. However, we can refer that the template in this case is learned and we can reduce the noises. Therefore, we can refer that this method has a good performance for simple binary images.

4.2. Applying Learned Template to Other Images

From this result, we obtain learned template which can do noise reduction. Next, we process other images using the learned template in order to confirm whether being able to apply learned template to other images or not.

Figure 2(a) shows the input image of CIRCLE. Figure 2(b) shows the desired image of CIRCLE. Figure 2(c) shows the output image using learned template. Figure 2(d) shows the input image of TRIANGLE. Figure 2(e) shows the desired image of TRIANGLE. Figure 2(f) shows the output image using learned template.

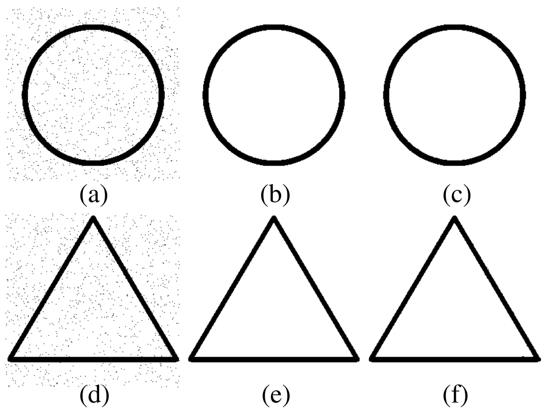
Figure 2: The images of simulation results. (a) Input image of CIRCLE. (b) Desired image of CIRCLE. (c) Output image of CIRCLE. (d) Input image of TRIANGLE. (e) Desired image of TRIANGLE. (f) Output image of TRIANGLE.

From these results, we can refer that CNN using learned template can remove the noise. Because we cannot distinguish output image using learned template from the desired image. When we use the learned template for similar images to first inputted image, CNN can process images correctly.

5. Conclusions

In this study, we modified the BP algorithm for CNN template design. As a result, we obtained learned template for two-dimensional image, and showed the performance of the proposed algorithm. However, in this study, we used simple binary images and simple tasks. As future works, we apply the proposed algorithm for more difficult tasks and use gray scale images.

References

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

[2] M. Nakagawa, T. Inoue and Y. Nishio, "CNN Template Design Using Back Propagation Algorithm," Proc. of CNNA'10, pp. 47-51, Feb. 2010.

[3] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning Internal Representations by Error Propagation," Parallel Distributed Processing, vol. 1, pp. 318-362, 1986.

[4] M. Nakagawa, Y. Uwate and Y. Nishio, "Template Design of CNN by BP with Annealing Noise," IEICE Tech. Rep., vol. NLP2010-67, & CAS2010-51, pp. 93-98, Aug. 2010 (in Japanese).