Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

Cellular Neural Network with Dynamic Template and its Output Characteristics

Masakazu KAWAHARA, Takashi INOUE and Yoshifumi NISHIO

Abstract— **In this research, we propose cellular neural network with dynamic template. In our proposed cellular neural network, the template of cell is changed at each update by learning. The learning method considering rank order learning is inspired from self-organizing map. Some computer simulations using the proposed cellular neural network with dynamic template are carried out and its output characteristic is investigated through simple image processing task.**

I. INTRODUCTION

CELLULAR Neural Networks (CNN) [1] were introduced by Chua and Yang in 1988. 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 application [2]-[4] including medical image processing [5][6] and texture segmentation [7]. Further, template learning is important subject in the studies of the CNN [8].

Wiring weights of the cells of the CNN are established by parameters called the template. The performance of the CNN is decided by the template. Usually, the templates of all the cells in the CNN are identical and those values do not change during the processing. This is good for implementation but restrict the performance, namely the conventional CNN can not perform image processing based on the local features of input images.

In this study, we propose a new type of CNN, whose template is dynamically updated by learning. We consider the learning method with rank order learning inspired from the self-organizing map (SOM) [9]. The SOM is a kind of neural networks modeling the visual area in brain. In the SOM, all neurons have state values as vectors and change their state values by input data. The learning process is based on the rank order learning and the winner neuron plays an important role. We investigate the basic output characteristic of the proposed CNN with dynamic template (D-CNN) by computer simulations.

In the Sec. 2, we review the basic of the standard CNN. In the Sec. 3, we show the algorithm of the proposed D-CNN. In the Sec. 4, simulation results using D-CNN are shown. In the Sec. 5, we consider the stability of D-CNN. In the Sec. 6, we show the complexity of output images obtained by D-CNN. In the Sec. 7, concludes the article.

Department of Electrical and Electronic Engineering, Tokushima University, 2-1 Minami-Josanjima, Tokushima 770-8506, Japan (email: {kawahara, takashi, nishio}@ee.tokushima-u.ac.jp).

II. CELLULAR NEURAL NETWORKS [1]

In this section, we describe the basic structure of the CNN. The CNN has M by N processing unit circuits called cells. 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 follows.

State equation:

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

Output equation:

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

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.

III. CELLULAR NEURAL NETWORK WITH DYNAMIC TEMPLATE (D-CNN)

In this section, we explain the algorithm of D-CNN. In our D-CNN, the template of cell is changed at each update by learning. The learning steps in our D-CNN are shown as follows.

STEP 1: The state values and the output values of all the cells in D-CNN are updated according to the discretized model of Eqs. (1) and (2).

STEP 2: Calculate the comparison of the output value of each cell with the one-step past outputs of the cell and its neighbor cells. The comparison equation for the cell $(i,j)$is described as follow.

Comparison Equation:

$$Dif(i,j;k,l) = |v^{now}_{y,(i,j)} - v^{past}_{y,(k,l)}| \tag{3}$$

This comparison corresponds to the function finding the winner neuron in SOM.

STEP 3: Among the 9 calculated values of $Dif(i,j;k,l)$, the cell with the smallest value is defined as "winner" and the cell with the second smallest value is defined as "second". By this step, we find the position of cells with the nearest and the second nearest values to the corresponding cell (i,j).

STEP 4: Update the elements of the template corresponding to the positions of the winner and the second. Note that in our proposed learning algorithm only two elements are updated. The update function is described as follows.

Update Method:

Assume that the template before update is given as Eq. (4).

$$Template^{now}:$$

$$A^{now}_{(i,j)} = \begin{bmatrix} a^{now}_{11} & a^{now}_{12} & a^{now}_{13} \\ a^{now}_{21} & a^{now}_{22} & a^{now}_{23} \\ a^{now}_{31} & a^{now}_{32} & a^{now}_{33} \end{bmatrix},$$
$$B^{now}_{(i,j)} = \begin{bmatrix} b^{now}_{11} & b^{now}_{12} & b^{now}_{13} \\ b^{now}_{21} & b^{now}_{22} & b^{now}_{23} \\ b^{now}_{31} & b^{now}_{32} & b^{now}_{33} \end{bmatrix},$$
$$I^{now}_{(i,j)} = I^{now}. \tag{4}$$

For example, we consider the case that the winner is (i,j) and the second is $(i-1,j-1)$. In that case, only a^{now}_{22}, b^{now}_{22}, a^{now}_{11} and b^{now}_{11} in Eq. (4) are updated. The threshold value I is not updated in this algorithm.

In our update algorithm, we change the learning rate in two elements. The learning rates of the winner and the second are shown as follows.

Learning rate:

$$Winner: \quad R_{10} \qquad (-0.1 \leq R_{10} \leq 0.1). \tag{5}$$
$$Second: \quad R_{20} = R_{10}/4 \ (-0.025 \leq R_{20} \leq 0.025). \tag{6}$$

Further, the learning rate decreases as time goes according to the following equation.

$$R_1 = R_{10}\left(1 - \frac{Number\ of\ calculation_{now}}{Number\ of\ calculation_{max}}\right). \tag{7}$$
$$R_2 = R_{20}\left(1 - \frac{Number\ of\ calculation_{now}}{Number\ of\ calculation_{max}}\right). \tag{8}$$

In this study, we decide $Number\ of\ calculation_{max}$ in Eqs. (7) and (8) is set to 10. Namely, the learning rates of winner and second are changed until 10 calculations. Then, after $Number\ of\ calculation_{max}$ becomes over 10, the learning rate of winner and second becomes 0 and the template is not updated anymore. This convergence of the learning rate is also inspired from the learning algorithm of SOM.

By using the learning rate, the elements of template are updated according to the following update equation.

Update Equation:

$$a^{updated}_{11} = a^{now}_{11} + R_2(v^{now}_{y,(i,j)} - v^{past}_{y,(i,j)}). \tag{9}$$
$$a^{updated}_{22} = a^{now}_{22} + R_1(v^{now}_{y,(i,j)} - v^{past}_{y,(i,j)}). \tag{10}$$

In Eqs. (9) and (10), $a^{updated}_{11}$ and $a^{updated}_{22}$ are the updated values. Also, b_{11} and b_{22} are updated similarly. After the update using Eqs. (9) and (10), the updated template is shown as follows.

$$Template^{updated}:$$

$$A^{updated}_{(i,j)} = \begin{bmatrix} a^{updated}_{11} & a^{now}_{12} & a^{now}_{13} \\ a^{now}_{21} & a^{updated}_{22} & a^{now}_{23} \\ a^{now}_{31} & a^{now}_{32} & a^{now}_{33} \end{bmatrix},$$
$$B^{updated}_{(i,j)} = \begin{bmatrix} b^{updated}_{11} & b^{now}_{12} & b^{now}_{13} \\ b^{now}_{21} & b^{updated}_{22} & b^{now}_{23} \\ b^{now}_{31} & b^{now}_{32} & b^{now}_{33} \end{bmatrix},$$
$$I^{updated}_{(i,j)} = I^{now}. \tag{11}$$

STEP 5: The steps from 1 to 4 are repeated.

IV. Simulation results

In order to investigate the basic property of the proposed D-CNN, we concentrate on the performance using diffusion template in this article. Usually, the diffusion CNN diffuses the input image without limit. Namely, all values of cells converge to one constant value. We consider that the values of all cells converge to some different points as clustering performed by SOM by virtue of the effect of the rank order learning.

Diffusion Template:

$$A = \begin{bmatrix} 0.1 & 0.15 & 0.1 \\ 0.15 & 0 & 0.15 \\ 0.1 & 0.15 & 0.1 \end{bmatrix}, \ B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$
$$I = 0. \tag{12}$$

Figure 1 shows the simulation results using diffusion template (12). Figure 1(a) is an input image. Figure 1(b) shows the simulation result using diffusion temperate (12) in the original CNN. We can see in Fig. 1(b) that the output image obtained by the original diffusion CNN converges to the image with uniform constant color. Figures 1(c)∼(f) show the simulation results using diffusion template (12) in D-CNN for different learning rates. In Figs. 1(c)∼(f), the output image is divided into two or three colors. Additionally, the output image has difference in the two area depending on whether if the learning rate is positive or negative. If the value of the learning rate is positive, the output image is divided into white area and gray area. While if the value of

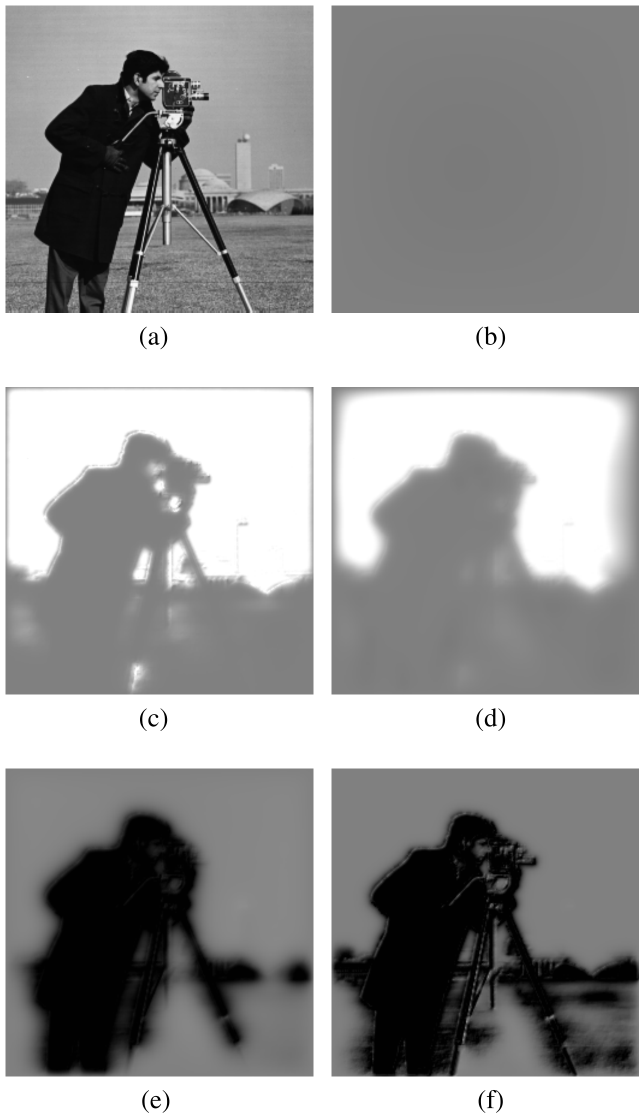


Fig. 1. Simulation results using diffusion template (12) in D-CNN. (a) Input image (Cameraman). (b) Diffusion result using diffusion template (12) in original CNN. (c) Output image using D-CNN ($R_{10} = 0.1$). (d) Output image using D-CNN ($R_{10} = 0.02$). (e) Output image using D-CNN ($R_{10} = -0.02$). (f) Output image using D-CNN ($R_{10} = -0.1$).

the learning rate is negative, we can see three types domains as black, gray, and the middle color of black and gray.

Then, we investigate the distribution of colors in the output image. Since the output value of cells is between -1 and 1, we divide the interval to 10 small segments and count the number of cells whose values are in each segment. Figure 2 shows the color density distributions. Figure 2(a) is the distribution of the original input image. Figure 2(b) is the distribution of the output image in Fig. 1(b) obtained by the original CNN . From this result, we can see that all the cells converge near 0. Figures 2(c) and (d) are the distributions of the output images in Figs. 1(c) and (f) obtained by the proposed D-CNN. From these results, we can see that many cells converge to two different values and that some cells converge to intermediate values.

Figure 3 shows the simulation results for different input image with the same diffusion template. Similar to the case of the previous results, the proposed D-CNN produces the color divided output images.

Similarly, Fig. 4 shows the color density distributions. Figure 4(a) is the distribution of the original input image. Figure 4(b) is the distribution of the output image in Fig. 3(b) obtained by the original CNN . We can see that almost cells converge near 0. Figures 4(c) and (d) are the distributions of the output images in Figs. 3(c) and (f) obtained by the proposed D-CNN. From these results, we can see that many cells converge to two different values and that some cells converge to intermediate values.

V. Stability of D-CNN

In this section, we consider the stability of D-CNN. From the simulation results in Figs. 1 and 3, we can see that the diffusion of the D-CNN divide the input images to two or three color regions. Then, we investigate the process of convergence by choosing some pixels of the images. Further, we compare the proposed D-CNN with the original CNN.

Figure 5 shows the graphs of convergence process when the input image is Cameraman. Three curves correspond to three different pixels whose positions are indicated in the box in each figure. Figure 5(a) is the convergence process graph of the original CNN. We can see that all three curves converge to around zero after long transient like 150000[s]. Figures 5(b) and (c) are convergence process graphs when the learning rate is positive and negative in D-CNN, respectively. From Figs. 5(b) and (c), the convergence time in our D-CNN is much more rapid (less than 1000[s]) than that of the original CNN. Also, the curves converge to different values.

Figure 6 shows the graphs of convergence process when the input image is Sailboat. Similar to Fig. 5, Fig. 6(a) shows the convergence of the original CNN, while Figs. 6(b) and (c) show the convergences of D-CNN with positive and negative learning rates, respectively. We can see that the similar features are confirmed in this simulation result.

VI. Complexity of output images

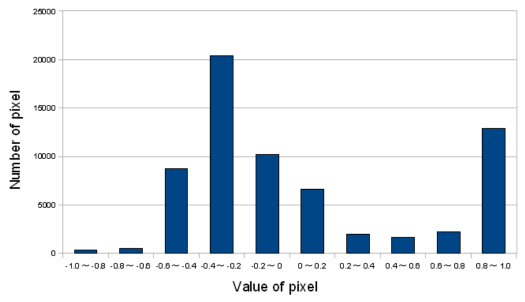
In this section, we examine the complexity of the output images obtained by the proposed D-CNN. If the value of the complexity is small, the density difference of the image is small. On the other hand, if the value of the complexity is large, the density difference of image is large, and we can say that the image is complex. Hence, we can use this index to evaluate the diffusion effect of CNN.

The complexity of images can be calculated as follows.

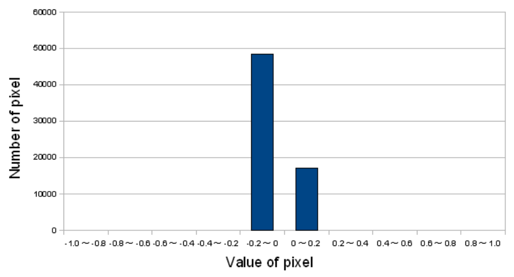
STEP 1: We calculate the differentiations of horizontal and vertical directions of the image by using the differential operators in Eqs. (13) and (14).

Differentiation of horizontal direction:

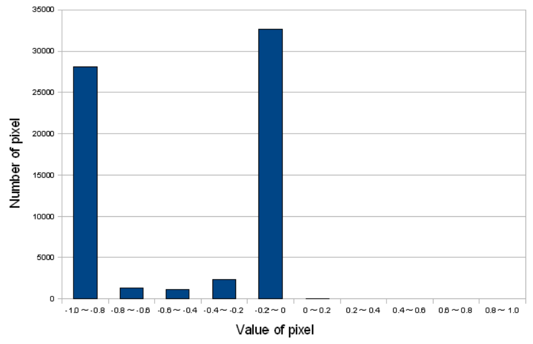
$$\frac{\partial z}{\partial x} = \frac{1}{6}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{13}$$



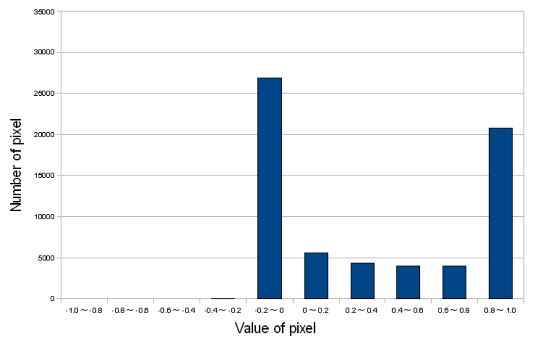
(a)



(b)

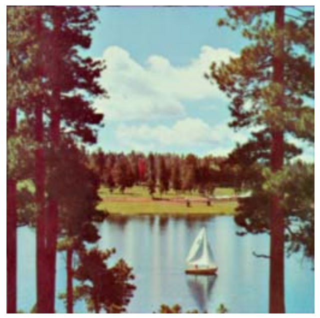


(c)

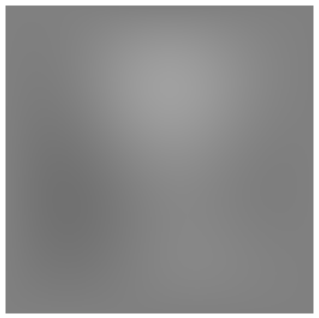


(d)

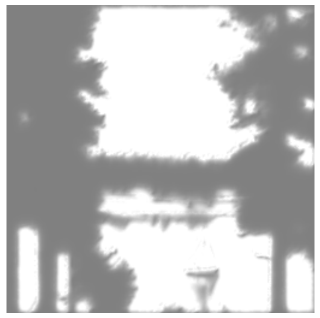
Fig. 2. Color density distribution of output image. (a) Input image (Cameraman). (b) Output image obtained by original CNN. (c) Output image obtained by D-CNN with $R_{10} = 0.1$. (d) Output image obtained by D-CNN with $R_{10} = -0.1$.



(a)

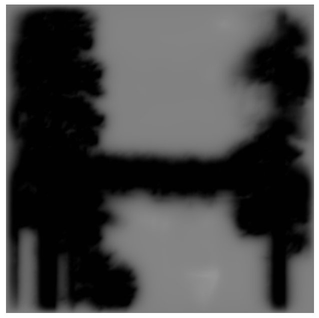


(b)

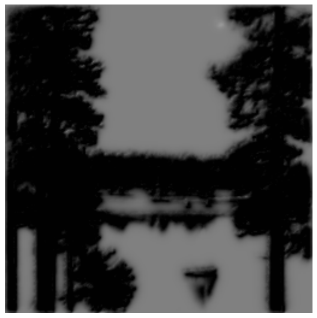


(c)

(d)



(e)



(f)

Fig. 3. Simulation results using diffusion template (12) in D-CNN. (a) Input image (Sailboat). (b) Diffusion result using diffusion template (12) in original CNN. (c) Output image using D-CNN ($R_{10} = 0.1$). (d) Output image using D-CNN ($R_{10} = 0.02$). (e) Output image using D-CNN ($R_{10} = -0.02$). (f) Output image using D-CNN ($R_{10} = -0.1$).

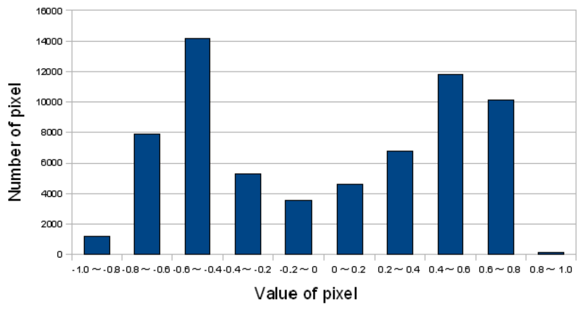
Differentiation of vertical direction:

$$\frac{\partial z}{\partial y} = \frac{1}{6}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{14}$$

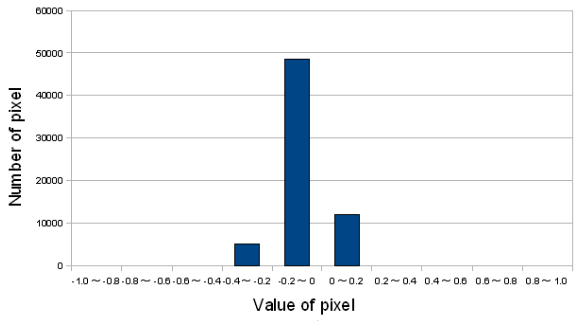
STEP 2: According to the following equations, we combine the differentiations Eqs. (13) and (14)

Calculate Equation 1:

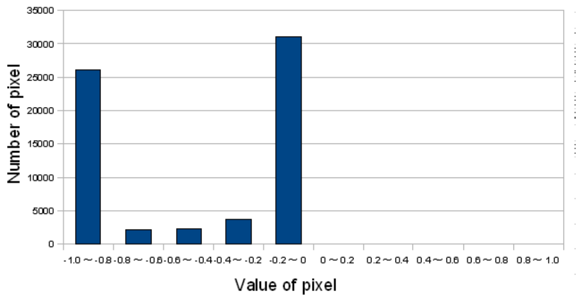
$$D_{11} = \frac{\partial^2}{\partial x^2} \tag{15}$$



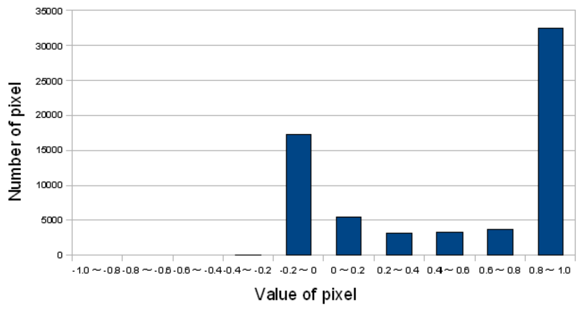
(a)



(b)

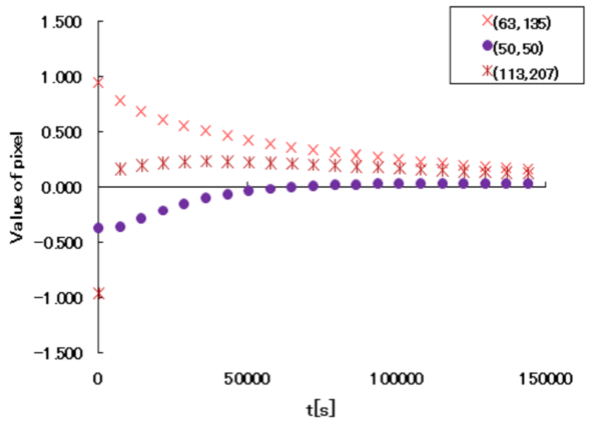


(c)

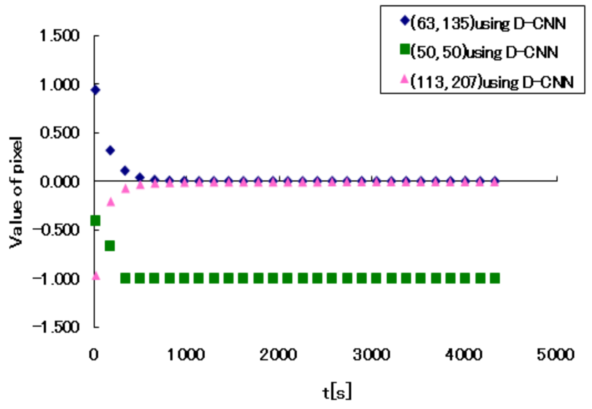


(d)

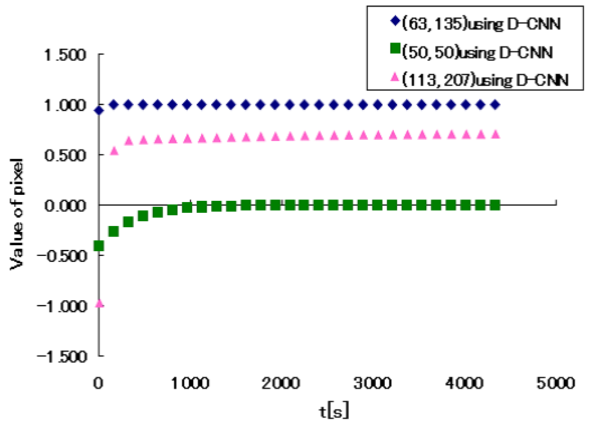
Fig. 4. Color density distribution of output image. (a) Input image (Sailboat). (b) Output image obtained by original CNN. (c) Output image obtained by D-CNN with $R_{10} = 0.1$. (d) Output image obtained by D-CNN with $R_{10} = -0.1$.



(a)

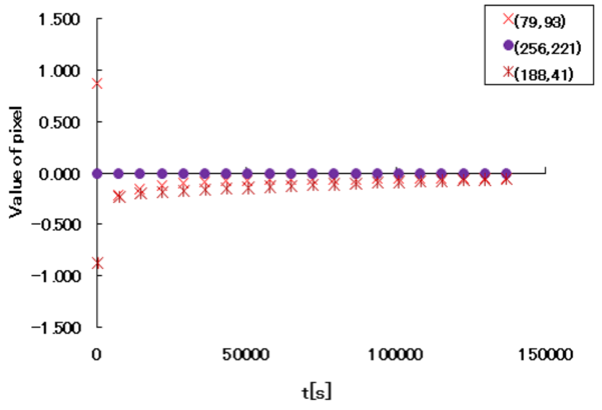


(b)

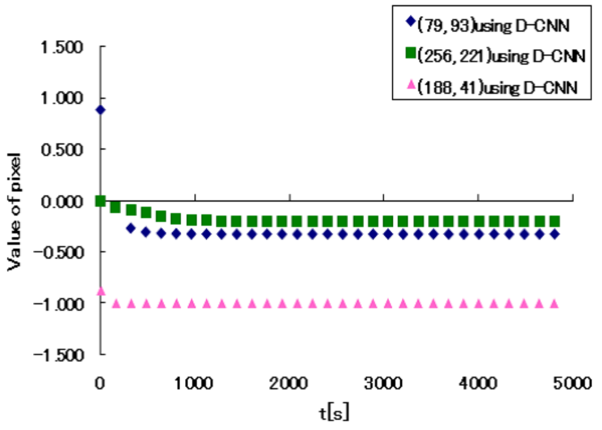


(c)

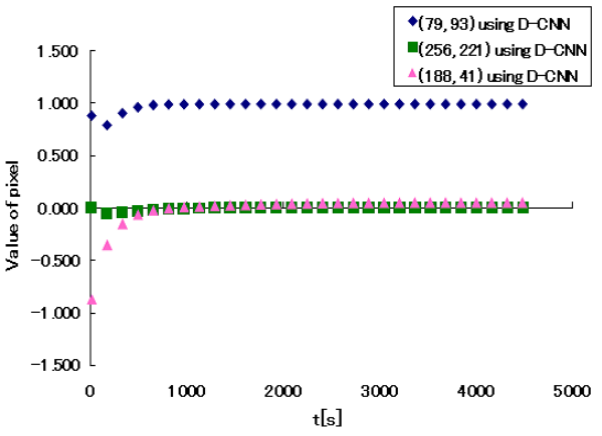
Fig. 5. Convergence process for Cameraman. (a) Original CNN. (b) D-CNN with $R_{10} = 0.1$. (c) D-CNN with $R_{10} = -0.1$.



(a)



(b)



(c)

Fig. 6. Convergence process for Sailboat. (a) Original CNN. (b) D-CNN with $R_{10} = 0.1$. (c) D-CNN with $R_{10} = -0.1$.

Calculate Equation 2:

$$D_{12} = \frac{\partial^2}{\partial x \partial y} \tag{16}$$

Calculate Equation 3:

$$D_{21} = \frac{\partial^2}{\partial y \partial x} \tag{17}$$

Calculate Equation 4:

$$D_{22} = \frac{\partial^2}{\partial y^2} \tag{18}$$

STEP 3: We calculate the complexity of a certain cell (i, j) by using Eq. (19).

Combine Equation:

$$fin(i,j) = (D_{11}(i,j))^2 + (D_{12}(i,j))^2 + (D_{21}(i,j))^2 + (D_{22}(i,j))^2 \tag{19}$$

STEP 4: Finally, we calculate the complexity of image by using Eq. (20). Namely, one image have one complexity value.

Complexity of Image Equation:

$$Fin = \sum_{i=0,j=0}^{i=255,j=255} fin(i,j) \tag{20}$$

Figure 7 shows the graph of complexity for Figs. 1 and 3 at various different learning rates. In Fig. 7, we can see that the complexity of the output image is low when the learning rate is near 0. Namely, when the learning rate is near 0, the output image of D-CNN becomes simple.

VII. Conclusions

In this research, we have proposed the cellular neural network with dynamic template (D-CNN). In D-CNN, the template of cell was changed at each update by learning. The learning method considering rank order learning was inspired from SOM. We investigated output characteristics of the proposed D-CNN through the diffusion image processing.

References

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

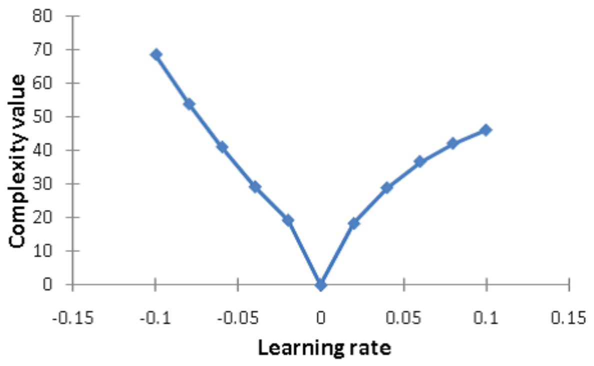
[2] F. Dirk and T. Ronald, “Coding of Binary Image Data using Cellular Neural Networks and Iterative Annealing,” Proc. of ECCTD’03, vol. 1, pp. 229-232, Sep. 2003.

[3] M. Namba and Z. Zhang, “Cellular Neural Network for Associative Memory and Its Application to Braille Image Recognition,” Proc. of IJCNN’06, pp. 4716-4721, Jul. 2006.

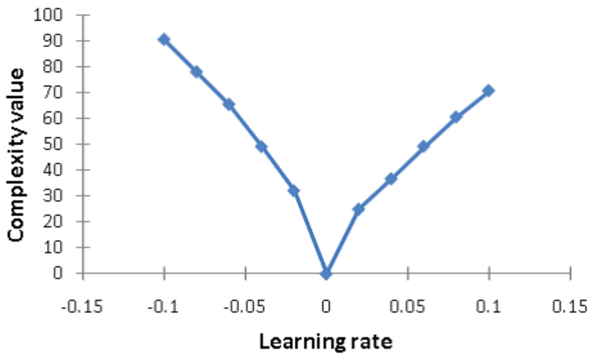
[4] H. Koeppl and L.O. Chua, “An Adaptive Cellular Nonlinear Network and its Application,” Proc. of NOLTA’07, pp. 15-18, Sep. 2007.

[5] R. Perfetti, E. Ricci, D. Casali, and G. Costantini “Cellular Neural Networks With Virtual Template Expantion for Retinal Vessel Segmentation,” IEEE Trans. Circuits Syst., vol. 54, pp. 141-145, Feb. 2007.

[6] T. Szabó and P. Szolgay, “An Analogic Diffusion-Based Algorithm for the Segmentation of CT Images,” Proc. of NOLTA’07, pp. 237-239, Sep. 2003.



(a)



(b)

Fig. 7. Complexity of output image. (a) Output image for Cameraman. (b) Output image for Sailboat.

[7] C.T. Lin, C.H. Huang and S.A. Chen, "CNN-Based Hybrid-Order Texture Segregation as Early Vision Processing and Its Implementation on CNN-UM" IEEE Trans. Circuits Syst., vol. 54, pp. 2277-2287, Oct. 2007.

[8] C.L. Chang, K.W. Fan, I.F. Chung, and C.T. Lin, "A Recurrent Fuzzy Coupled Cellular Neural Network System With Automatic Structure and Template Learning" IEEE Trans. Circuits Syst., vol. 53, pp. 602-606, Aug. 2006.

[9] T. Kohonen, *Self-Organizing Maps*, 2nd ed., Berlin, Springer, 1995.