CNNA-94 Third IEEE International Workshop on Cellular Neural Networks and their Applications
Rome, Italy, December 18-21,1994

Texture Classification, Texture Segmentation and Text Segmentation with Discrete-Time Cellular Neural Networks

Andreas Kellner, Holger Magnussen and Josef A. Nossek
Institute for Network Theory and Circuit Design,
Technical University Munich, Munich, Germany
e-mail: holgi@nws.e-technik.tu-muenchen.de

Abstract – *Global Learning Algorithms presented in a companion paper are applied to practical classification and segmentation problems: Texture Classification and Texture Segmentation of artificial and natural textures, and Text Segmentation as a sub-problem of Page Layout Analysis. In all cases, DTCNN systems can solve the problem very well in spite of its only local interconnection structure.*

1 Introduction

The Discrete-Time Cellular Neural Network (DTCNN) [1] is a discrete-time, nonlinear, first-order dynamical system consisting of M identical cells on a 1D or 2D cell grid. It is defined by the equations

$$\begin{aligned} x_\mu(t+1) &= \sum_{\lambda\in\mathcal{N}(\mu)} a_{\lambda-\mu} y_\lambda(t) + \sum_{\lambda\in\mathcal{N}(\mu)} b_{\lambda-\mu} u_\lambda(t) + i \\ y_\mu(t) &= \mathrm{SGN}(x_\mu(t)) := \begin{cases} +1 & \text{for } x_\mu(t) \geq 0 \\ -1 & \text{for } x_\mu(t) < 0 \end{cases} \\ y_\mu(0) &= \mathrm{SGN}\left(\sum_{\lambda\in\mathcal{N}(\mu)} b^0_{\lambda-\mu} u_\lambda(0) + i^0\right) . \end{aligned} \tag{1}$$

$u_\mu(t)$, $y_\mu(t)$, and $x_\mu(t)$ are the input signal at port μ, the output signal and the state of cell μ, respectively, at time-step t. $\mathbf{a} = (a_\nu)$ and $\mathbf{b} = (b_\nu)$ are the feedback and feedforward template coefficients, i is the cell bias. $\mathbf{b}^0 = (b_\nu)^0$ and i^0 are an additional template and an additional bias for the computation of the initial state $y_\mu(0)$. $\mathcal{N}(\mu)$ is the *neighborhood* of cell μ on the cell grid. The cell grid is surrounded by r layers of *dummy cells*.

Learning is achieved by means of the two global learning algorithms presented in a companion paper [2]. The goal of these learning algorithms is to find the template parameters so that the network maps a set of input images onto a set of corresponding desired output images. The above learning algorithms were applied to the fields of Texture Classification (Section 2), Texture Segmentation (Section 3), and Text Segmentation (Section 4).

2 Texture Classification

Texture classification and texture segmentation are two fundamental preprocessing steps in image analysis. In the texture classification task, an input image has to be classified as belonging to one of several previously known texture classes. Most of the traditional systems require several steps (cf. [3], [4], [5]): First, a feature matrix is computed from the original image (e.g. using FFT or Hough-Transform). In the next step, a feature vector is extracted from this matrix. This vector is then used in the clustering process to determine the correct texture class.

In our approach, all these steps are replaced by a single DTCNN, which determines the texture class directly from the input image.

A standard DTCNN with a 1-neighborhood in the cell grid and a 2-neighborhood in the input grid was used for this problem. The dummy cells of the cell grid were set to zero, and the dummy cells of the input grid were set to the value of the nearest active input cell.

The *gray level* $g(t)$ denoting the fraction of black pixels in the output pattern was used as output signal of the DTCNN:

$$g(t) = \frac{1}{2M} \sum_{\mu=1}^{M} (y_\mu(t) + 1) \,. \tag{2}$$

The signal g^∞ denotes the gray level of the stable output of the DTCNN. If the trajectory reaches a stable limit cycle, the average value of $g(t)$ over one period of the cycle is used.

If only two different texture classes have to be distinguished, a reasonable strategy is to map input images of one class onto $g^\infty = 0$, and input images of the other class onto $g^\infty = 1$. Our experiments have shown that the HYBRID algorithm can find appropriate templates for different pairs of texture classes. If more than two texture classes have to be classified, other strategies have to be used. The two following strategies work particularly well:

2.1 Self Organizing Gray Level Method

One approach is to map the K different texture classes onto different gray levels. This strategy is similar to the one used in [6]. Each texture class k has a gray level target g^k associated with it. By choosing a special form for the objective function, the targets are found by the learning algorithms as well. The goal for the objective function is to obtain an optimal separation between the targets g^k corresponding to different texture classes, and at the same time a minimal variance of the network responses g^∞ belonging to one texture class. Let $\mathbf{p}$ denote the parameter vector.

Then we obtain for the objective function $o(\mathbf{p})$:

$$o(\mathbf{p}) = \left(\sum_{k=1}^{K} \sigma_k \right) \cdot \left(\sum_{k=1}^{K-1} \sum_{j=k+1}^{K} \frac{1}{\|g^k - g^j\|} \right) , \tag{3}$$

where $\bar{g}_k$ is the average gray level of all input patterns of class k and σ_k is the variance of the gray levels in this class.

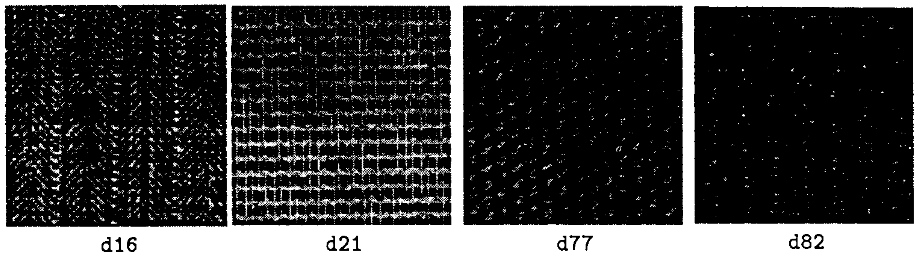

Fig. 1: Natural textures

A training set consisting of 20 natural textures from [7] was created. The input images with a 40×40 pixel size were taken from 4 different texture classes and normalized to zero mean and identical standard deviation (see Fig. 1). The Learning algorithms found a template that mapped the input images onto gray levels as shown in Fig. 2. Obviously, a perfect separation of the texture classes in the training set is possible.

The template was then applied to several test sets each consisting of 40 images from the same texture classes. The images could be classified very well. The system was even capable of classifying input images that were slightly scaled or skewed. In theses cases however, some input images were misclassified (see Fig. 3).

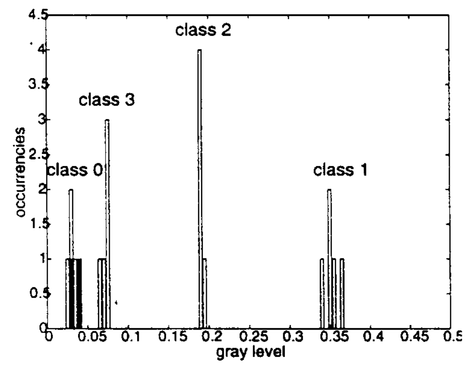

Fig. 2: Gray level Histograms for input images from the training set

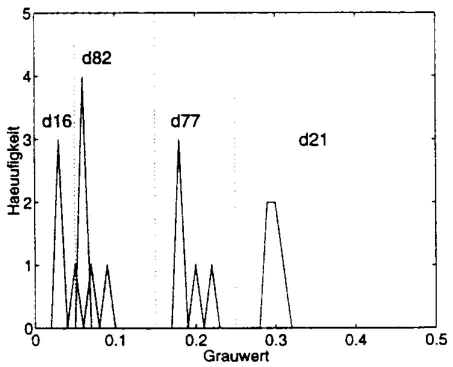

Fig. 3: Gray level histogram for input images rotated by 5°

2.2 1-in-K method

In the second approach, K DTCNNs using a 1-out-of-K coding were used to classify the K different texture classes. For each texture class, there is a DTCNN mapping this class onto $g^\infty = 0$ and all other classes onto $g^\infty = 1$. The input images are fed into all K DTCNNs. Two different recall modes were used: In the *Minimum Output* mode, the network with the smallest output value determines the class of the input pattern. In the *Coded Output* mode, the outputs of all networks are thresholded at $g_{\text{thr}} = 0.5$. A decision is only made if one network produces a logical "0" output and all other networks produce the logical output "1". In all other cases, the input pattern is rejected.

Templates for the $K = 4$ DTCNNs were found by the global learning algorithms, resulting in an error-free classification of the training set. A test set containing 40 different images was created. In the minimum output mode 97.5% of the patterns were classified correctly. In the coded output mode, 7.5% of the input patterns were rejected but no pattern was misclassified. Fig. 4 shows the outputs of the 4 DTCNNs (upper row: DTCNNS for the detection of d16 and d21, lower row: detection of d77 and d82) for an input image d77 from the training set. Test sets with slightly scaled input images were tested as well, resulting in a somewhat higher error rate.

Fig. 4: Output of the 4 DTCNNs of the 1-in-K method

3 Texture Segmentation

In the next step, the DTCNN 1-in-K system as described above was used for the more difficult task of texture segmentation, where the input images contain more than one texture class. Learning was done using a training set with multi-texture images. Again, the input images were fed into all $K = 4$ DTCNNs. A direct application of the Coded Output mode on a pixel level is too coarse, because in many cases near the boundaries between textures more than one network responds by a logical "0".

Since the output of a single DTCNN cell is binary-valued, the gray level of a single pixel in the DTCNN output was defined as the number of black pixel within a small neighborhood $\mathcal{N}_o$ (radius $r_o = 2$) around this pixel. The class of a pixel was then determined using the Minimum Output mode.

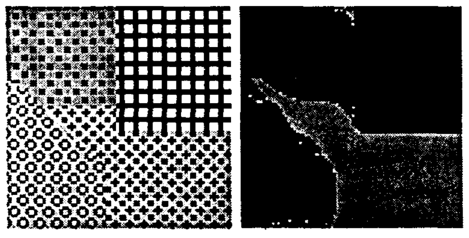
(a) (b)

Fig. 5: Segmentation of artificial textures: (a) input image (b) segmentation result

A first training set consisted of 4 classes of artificial textures with normalized means. Fig. 5 shows an input image from the test set and the resulting classification by the network.

Other experiments with natural textures from [7] were performed, and the results were only slightly worse than in the case of the artificial textures.

The *Self Organizing Gray Level Method* was also applied to the texture segmentation task. In this case, the average gray levels $\bar{g}_k$ and the variances σ_k used in the objective function (3) were computed pixel by pixel. The results for artificial textures were comparable to the *1-in-K-method.* A satisfactory segmentation of natural textures however was not possible with this method.

4 Text Segmentation

An important early processing step in automatic text reading systems is to determine, which parts of the page contain text, images, or graphics. These different regions are then further processed by specific algorithms. In this section, a DTCNN is used to detect and mark text blocks in a document. This is a part of the Layout Analysis problem, and it can be seen as a special case of a Texture Segmentation problem.

A page from a journal was scanned as an 8 bit greyscale image with a 52 DPI resolution (image size 514 × 386 pixel, see Fig. 6a). A training set consisting of 20 images with a 40 × 40 pixel size was created by segmenting images of this size from Fig. 6a (marked by boxes). The desired output images for the training set were designed manually (black for text regions, white otherwise).

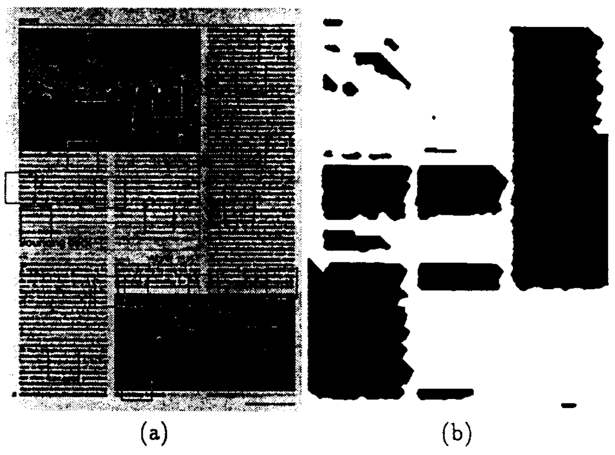

Fig. 6: Text Segmentation: (a) Input image with boxes denoting the training set; (b) output image

A DTCNN with zero dummy cells, with dummy ports having the value of the nearest active ports, and with 2-neighborhoods for all templates was used (77 independent parameters). The HYBRID method [2] was used during the learning phase. Templates could be found that solved the learning problem reasonably well, although the boundaries of the regions containing text were not absolutely accurate. The best template obtained during different runs of the learning algorithms was then applied to the whole 514×386 pixel input image. The network reaches a stable limit cycle of length $T_{cyc} = 5$ after $T_{tran} = 169$ time-steps. Fig. 6b shows the average over one limit cycle. Taking into account that each cell of the network is only connected to a 5 × 5 region of the whole 514 × 386 image (0.012% of the total number of cells), the results are surprisingly good. Even the headings of the page and the article and the captions of the photographs on the page are marked, although a different font is used in these parts of the document. A closer examination of the areas erroneously marked in the upper photograph

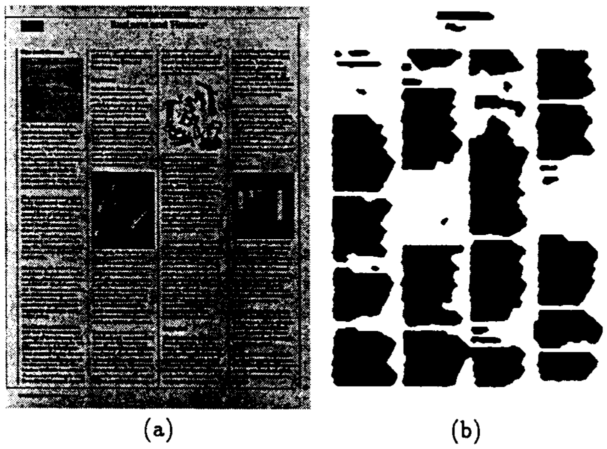

Fig. 7: Text Segmentation test example: (a) Input image; (b) output image

showed that the text printed on the back side of the page can still be seen through the paper. These faint traces of text are also detected by the network.

The system was tested by applying the same templates to the image shown in Fig. 7a, which uses the same font, but does not have fully justified text blocks and two additional bar graphs and a cartoon on the page. The stable result after $T_{tran} = 121$ time-steps is shown in Fig. 7b. Note that parts of the cartoon are erroneously marked as text blocks, and that the system is even able to detect the captions inside the shaded boxes. Application of the DTCNN with the same templates to a slightly tilted version of image Fig. 6a and a page from a Russian journal with a cyrillic font resulted in equally good results. When compared to the results of typical higher level approaches [8], [9], [10], the output of the DTCNN is still relatively coarse. However, the proposed DTCNN Texture Segmentation application has large advantages with respect to processing speed and parallelizability.

5 Acknowledgements

The authors would like to thank Professor L.O. Chua for enabling the research stay of Andreas Kellner at the Electronics Research Laboratory, University of California, Berkeley, where the work on Texture Classification and Texture Segmentation was done.

References

[1] H. Harrer and J. A. Nossek, "Discrete-Time Cellular Neural Networks," *International Journal of Circuit Theory and Applications*, vol. 20, pp. 453–467, Sept. 1992.

[2] H. Magnussen and J. A. Nossek, "Global learning algorithms for discrete-time cellular neural networks *(to appear)*," in *Proc. Third IEEE International Workshop on Cellular Neural Networks and their Applications CNNA-94*, (Rome, Italy), Dec. 1994.

[3] L. V. Gool, P. Dewaele, and A. Oosterlinck, "Texture analysis anno 1983," *Computer vision, Graphics, and Image Processing*, vol. 29, pp. 336–357, 1985.

[4] T. R. Reed and J. H. du Buf, "A review of recent texture segmentation and feature extraction techniques," *CVGIP: Image Understanding*, vol. 57, pp. 359–372, Mar. 1993.

[5] R. M. Haralick and L. G. Shapiro, "Image segmentation techniques," *Computer vision, Graphics, and Image Processing*, vol. 29, pp. 100–132, 1985.

[6] T. Szirányi and M. Csapodi, "Texture detection by cellular neural networks using genetic learning," Tech. Rep. DNS-8-1993, Analogic and Neural Computing Research Laboratory, Computer and Automation Institute; Hungarian Academy of Sciences (MTA SzTAKI), 1993.

[7] P. Brodatz, *Textures; A Photographic Album for Artists and Designers.* Dover Publications, 1966.

[8] A. Jain and S. Bhattacharjee, "Text segmentation using gabor filters for automatic document processing," *Machine Vision and Applications*, vol. 5, no. 3, pp. 169–184, 1992.

[9] G. Nagy, S. Seth, and M. Viswanathan, "A prototype document image analysis system for technical journals," *Computer*, vol. 25, pp. 10–22, July 1992.

[10] S. Tsujimoto and H. Asada, "Major components of a complete text reading system," *Proceedings of the IEEE*, vol. 80, pp. 1133–1149, July 1992.