An Assembly of SSO Neural Networks for Character Recognition

Chiung-Yao Fang†, Chih-Ping Tung†, Sei-Wang Chen†, and Chiou-Shann Fuh‡

†Department of Information and Computer Education
National Taiwan Normal University
Taipei, Taiwan
Email: violet@ice.ntnu.edu.tw
‡Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan

Abstract

This paper presents a method to recognize the characters printed on integrated circuits (ICs). Using one kind of features extracted from these characters, we could train a spherical self-organizing (SSO) neural network to form a feature map. Since there are several feature maps to be constructed, an assembly of SSO neural networks is introduced to recognize these alphanumeric characters. The experimental results show the high accuracy and robustness of the system.

I. Introduction

Character recognition is useful in many applications, such as identification of private seals, inspection of printed integrated circuit (IC) labels, document analysis, as well as automatic reading systems. Various character recognition techniques [Bus 97], [Cao 97], [Cho 97], [Mar 97], [Sri 97], [Web 98], have been available and are characterized by structural, statistic, evidence theoretic, neural, and fuzzy approaches. In this study, an assembly of spherical self-organizing (SSO) neural networks is introduced to recognize 36 alphanumerical characters, including 0,…, 9, and A,…, Z.

Experimental characters are extracted from labels of IC chips. There are difficulties encountered with the extracted characters. First, characters might be incomplete or connected due to imperfect segmentation of characters. Second, characters could be noisy because of degradation of input images. Third, the scale

change within a character varies from direction to direction. Fourth, there are distinct fonts associated with a character. Finally, several character pairs (e.g., 0 and O, 1 and I, as well as 5 and S) look similar to each other in shape. The proposed recognition technique can handle all these difficulties.

Neural networks have been well known to possess a number of intriguing properties suitable for pattern classification and recognition. Self-organizing (SO) feature maps, introduced by Kohonen [Koh 89], are one of such neural networks. There are distinguished characters with the SO neural networks. Of which, the most important one may be the topology-preserving projection. It is this property that the Kohonen SO neural model becomes adequate for pattern classification. Moreover, we introduced a SSO neural model to avoid some shortcoming of the Kohonen SO neural model.

In our paper, there are nine features [Rom 97] computed from a character for both the purposes of representation and recognition. Rather than input at a time all the features extracted to a single SSO neural network, a multitude of neural networks each dedicated to a feature are integrated. Therefore, there are nine SSO neural networks connected in parallel associated with the nine features in the proposed system. The output of each neural network is a list of possible classes for the input character. There is a value accompanying each class reflecting the neighborhood order of the input character to the class. In decision, the input character will be regarded as the character corresponding to the class having the largest summed neighborhood value over all the lists. The experimental results show that our system reliably recognized all the characters.

II. The SSO Neural Network

The Kohonen SO neural model [Koh 89] has been configured as a two-layer network: one input layer and one output layer. The output layer is more often referred to as an SO layer. On this layer, feature maps, inspired by the significant configuration of neurons in the cerebral cortices of biological brains, are realized in terms of intrinsic status of the constituent neurons. During formation of a feature map, neurons on the SO layer compete with one another for each input stimulus. The winner and its neighbors within a range then participate in a learning process. The SO learning mechanism produces a localized neural response to the input stimulus. If innervation continues, clusters of responses will be established. The collection of clusters forms what we call a feature map. The map is meaningful in the sense that it is generated in

response to the significance of outside stimuli. This process has been referred to as the topology-preserving projection.

The SO layer is generally structured as a planar layer. There are several shortcomings with a planar SO layer. First of all, it is difficult to locate neighbors for boundary neurons of the layer, especially for those situated on corners, once they become winners. A spherical SO layer, as Figure 1 shows, will avoid this dilemma in view of perfectly symmetrical construct. Furthermore, the closed structure of the spherical layer also contributes to the convergence rate
in the course of training.

A. Fundamental Mechanism

SO neural networks possess a number of key features, including lateral interaction, group learning, and clustering. During operation, neurons on the output layer compete with one another for input stimuli. The winner in each competition is initiated through locally *lateral inhibitio*n among output neurons. Both the winner and its neighbors to some extent join in a learning process. This process progressively localizes neural responses after repeated innovations. The output layer eventually stabilizes at a pattern of neural responses. The process of localizing neural responses has been referred to as *clustering* and the formation of response pattern as the *topology-preserving projection*.

B. Mathematical Formulation

Suppose that the input layer of the neural network consists of *m* neurons and the output layer comprises *n* neurons. Let w_{ij} denote the strength of the link between output neuron *i* and input neuron *j*. The *strength vector* of output neuron *i* is written as $w_i = (w_{i1}, w_{i2},\ldots,w_{im})$. The input to output neuron *i* due to innervation *y* is defined by

$$I_i^v = w_i \cdot y = \sum_{k=1}^{m} w_{ik} \cdot y_k$$

The lateral interaction among output neurons is characterized by a "Mexican-hat" function, denoted by M(r), where r is the position vector of a neuron from the winner. The "Mexican-hat" function is often approximated by the Laplacian of Gaussian $\nabla^2 g(r)$ or the difference of Gaussians $g_1(r)$-$g_2(r)$.

Let u_{ik} be the synaptic strength of the connection between neurons *i* and *k* with the respective positions r_i and r_k. The input to output neuron *i* due to lateral interaction can be formulated as,

$$I_i^l = \sum_{k \in N, k \neq i} [u_{ik} \cdot M(r_k - r_i) \cdot a_k],$$

where N is the set of output neurons and a_k is the activation of neuron k. The transfer function of output neurons is defined below

$$\Psi(x) = \frac{1}{2}[1 + \tanh(-\lambda)x)],$$

where parameter λ controls the steepnesses of function landscapes. The larger λ is the steeper function landscapes are, and vice versa.

A leakage term is introduced for each output neuron which will dissipate activations of neurons once stimuli have been removed. Let $e(a)$ denote this term. We obtain the net input to output neuron i as

$$net_i = I_i^v + I_i^l + e(a_i) = \sum_{k=1}^{m} w_{ik} \cdot y_k + \sum_{k \in N, k \neq i} [u_{ik} \cdot M(\mathbf{r}_k - \mathbf{r}_i) \cdot a_k] + e(a_i),$$

During competition, the winner, c, is defined by $net_c = \max_{i=1}^{n} \{net_i\}$. The winner and its neighbors participate in a learning process. The neurons within the neighborhood of the winner have differed in degree of learning governed by a Gaussian function.

$$G(\mathbf{r}) = \frac{1}{\sqrt{2\pi}} e^{-\|\mathbf{r}\|^2 / 2\sigma^2(t)},$$

where $\sigma(t)$ is the standard deviation of the neighborhood defined by the Gaussian.

Let N_c be the set containing the winner and its neighbors. The learning rule for the neurons in N_c is dictated by,

$$\Delta \mathbf{w}_k^z = \rho(z) \cdot (\mathbf{y} - \mathbf{w}_k^z) \cdot G(\mathbf{r}_k - \mathbf{r}_c).$$

The term $(\mathbf{y} - \mathbf{w}_k^z)$ in the above equation shifts strength vector $\mathbf{w}_k^z$ toward innervation $\mathbf{y}_j$. Function $G(\mathbf{r}_k - \mathbf{r}_c)$ imposes a degree of learning upon $(\mathbf{y} - \mathbf{w}_k^z)$ according to the distance $\mathbf{r}_c - \mathbf{r}_k$ of neuron k from the winner c. The updating equation for neuron k is then defined as,

$$\mathbf{w}_k^{z+1} = \mathbf{w}_k^z + \Delta \mathbf{w}_k^z, \qquad k \in N_c.$$

C. Processing Summary

The computational process of SO neural networks can be sketched as follows.

1. Initialize strength vectors $\mathbf{w}_i$ and activations a_i of output neurons.

2. Input an innervation **y**.
3. For each output neuron i, compute its net input by Eq. (3) and activation by $a_i = \Psi(net_i)$,
4. Determine the winner *c* by $net_c = \max_{i=1}^{n} \{net_i\}$.
5. Let N_c contain the winner and its neighbors; update the strength vectors of neurons in N_c according to Eq. (7).
6. Repeat Steps (2) to (5) until the strength vectors of output neurons stabilize.

III. Feature Extraction

Given an arbitrary-size image of a character, our system could binarize the image, remove the noises, and scale the size of image to 64×64 pixels. After these preprocessing steps, nine features described in Romero *et al.* [Rom 97] are computed as follows:

1. Stroke width and total stroke length

One binary character image could be convolved by three bitmap masks, as Figure 2 shows. Using anyone of the masks to scan every location in the image, the location could be marked if they match. If the number of foreground pixels is *b*, and the number of marked locations is *f*, then *b*/(*b-f*) and (*b-f*) could be computed as the average stroke width and the total stroke length. Thus, the feature is a two dimensional vector.

2. Horizontal and vertical projection histograms

Since the image has been scaled to 64×64 pixels, the horizontal and vertical projection histograms can be computed as a 128 (=64×2) dimensional feature vector.

3. Transitions

Counting the total number of transitions from background to foreground pixels along horizontal scan-lines, we could get one value of this two dimensional feature vector. The other value could be computed by vertical scan.

4. Peripheral features

Since the image could be divided into eight ``strips'' in both horizontal and vertical directions, the first-order (second-order) peripheral feature for a particular stripe is defined as the distance from the edge of strip to the first (second) background to foreground transition. The first and second-order peripheral features are computed from four directions---North, South, East, and West. Figure 3 shows an example of the first and second-order in the ``East'' direction. The first order peripheral feature is a 32

(=8×4) dimensional feature vector, so is the second order peripheral feature.

5. Stroke density

Counting the number of background to foreground transitions within each of eight horizontal and vertical strips as one dimensional value, the stroke density could be computed as a 16 (=8×2) dimensional feature vector.

6. Local direction contributivity

As Figure 4 shows, local direction contributivity locally measures the size of strokes in four orientations: 0°, 45°, 90°, and 135°. The image could be divided into 16×16 subimages by a 4×4 grid. For each orientation, we can compute the average length of the foreground line segments passing through each foreground pixel in each subimage. This feature is a 64 (=4×4×4) dimensional vector.

7. Stroke proportion

Stroke proportion is a proportional measure of four stroke directions: 0°, 45°, 90°, and 135°. The image could be divided into four strips for both horizontal and vertical directions. For each foreground pixel, its direction would be decided as the direction of the longest continuous foreground line passing through the pixel. For each strip, we can count the number of foreground pixels which belong to each direction, and divide it by the total number of foreground pixels in the strip. The feature is a 32 (=4×4×2) dimensional vector.

8. Maximum local direction contributivity

If the image is divided into blocks as local direction contributivity, and the direction of each foreground pixel is defined as stroke proportion, then, for each block, the number of foreground pixels belonging to each direction could be calculated. The feature is a 64 (=4×4×4) dimensional vector.

9. Black jump distribution in balanced subvectors of a character

If the 2D image is transformed to a 1D vector in each orientation: 0°, 45°, 90°, and 135°, we can divided the vector into 8 sub-vectors and count the number of background to foreground transitions in each vector. One example of the transformation is as Figure 5 shows. The feature is a 32 (=8×4) dimensional vector.

IV. The Assembly of SSO Neural Network

In our paper, there are nine features [Rom 97] computed from a character for

both the purposes of representation and recognition. Rather than input at a time all the features extracted to a single SSO neural network, nine neural networks each dedicated to a feature are assembled. Referring to Figure 6, every neural network preserves a classification of characters on its feature map prebuilt by training on the basis of a particular feature. Figure 7 shows an example of one SSO feature map. Therefore, there are nine SSO neural networks connected in parallel associated with the nine features in the proposed system.

The output of each neural network is a list of possible classes for the input character. There is a value accompanying each class reflecting the neighborhood order of the input character to the class. In decision, the input character will be regarded as the character corresponding to the class having the largest summed neighborhood value over all the lists.

V. Experimental Results

Recall that there are 36 distinct classes of characters. The recognition of a character will be accomplished by categorizing the input character into one of the 36 classes. Before the production phase, the system has to be trained first. For this purpose, there are only 36 character samples used for training, shown in Figure 8. For each character sample, nine features are computed. They are then used to train the nine SO neural networks in the system, respectively (n=9). At the end of training, a feature map is established in each SO neural network.

Having trained the system, it is ready to distinguish characters by means of classification. Our system generates 436 blurred or occluded characters from the 36 character samples, shown in Figure 9. The blurring method is described in Romero *et al.* [Rom 97], and the parameters are shown in Figure 9. The total recognition rate is 97.69%. One of the experimental results is shown in Figure 12. There is only one incorrect recognition in the experiment. Besides, the total cases of incorrect recognition are shown in Figure 13. The experimental results show that our system reliably recognized all the characters, even the training set is very small.

VI. Concluding Remarks

This paper presents a method to recognize the characters printed on integrated circuits. There are nine features computed from each character. Using one kind of features extracted from these characters, an SSO neural network could be

trained. The SSO neural network introduced in this paper could avoid some shortcomings of Kohonen SO neural network, especially for the boundary problem. Since there are nine SSO neural networks to be trained, we introduce an assembly method to combine the neural network system. The experimental result shows that the system could recognize the characters successfully.

There are some future works needed to be studied, including how to increase the recognition rate and speed. By increasing the training set, we believe that the recognition rate may be increased, but it needs more training time. If we could analyze the dependency of the features and only reserve the independent features, it may decrease the number of feature maps and increase the recognition speed. Besides, if the fuzzy measure could be introduced to assemble the SSO neural networks, the recognition rate may be increased.

somstr.eps
Figure 1: The structure of the SSO neural network.

page2.eps
Figure 2: Bitmap masks.

page3.eps
Figure 3: An example of the (a) first and (b) second-order peripheral features in the ``East'' direction.

page4.eps
Figure 4: An example of the four lines that pass through the pixel, one for each of 0°, 45°, 90°, and 135°.

page9.eps
Figure 5: The transformation of an image into a vector for the 45° orientation.

ssonetwork.eps
Figure 6: The architecture of the proposed neural system.

page14.eps
Figure 7: An example of one SSO feature map.

training.eps

Figure 8: The training set of the 36 classes.

recog.eps
Figure 9: Some examples of the 12 testing sets.

table.eps
Figure 10: An example of the experiment results.

error.eps
Figure 11: The examples of incorrect recognition.

References

[Bus 97] B. M. F. Bushofa and M. Spann, "Segmentation and Recognition of Arabic Characters by Structural Classification," *Image and Vision Computing,* Vol. 15, pp. 167-179, 1997.

[Cao 97] J. Cao, M. Ahmadi, and M. Shridhar, "A Hierarchical Neural Network Architecture for Handwritten Numeral Recognition," *Pattern Recognition,* Vol. 30, pp. 289-294, 1997.

[Cho 97] S. B. Cho, "Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals," *IEEE Transactions on Neural Networks,* Vol. 8, pp. 43-53, 1997.

[Koh 89] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1989.

[Mar 97] A. Marcelli, N. Likhareva, and T. Pavlidis, "Structural Indexing for Character Recognition," *Computer Vision and Image Understanding,* Vol. 66, pp. 330-346, 1997.

[Rom 97] R. D. Romero, D. S. Touretzky, and R. H. Thibadeau, "Optical Chinese Character Recognition Using Probabilistic Neural Networks," *Pattern Recognition,* Vol. 30, pp. 1279-1292, 1997.

[Sri 97] S. N. Srihari, T. Hong, and G. Srikantan, "Machine-Printed Japanese Document Recognition, " *Pattern Recognition,* Vol. 30, pp. 1301-1313,

1997.

[Web 98] R. G. Webster and M. Nakagawa, “An Interface-Oriented Approach to Character Recognition Based on a Dynamic Model,” *Pattern Recognition*, Vol. 31, pp. 193-203, 1998.