International Journal of Neural Systems, Vol. 3, No. 4 (1992) 395–404
© World Scientific Publishing Company

A SINGLE-LAYER NEURAL NETWORK FOR PARALLEL THINNING*

REI-YAO WU†
Institute of Computer Science and Information Engineering
National Chiao Tung University, Hsinchu, Taiwan 300, Republic of China

and

WEN-HSIANG TSAI‡
Department of Computer and Information Science, National Chiao Tung University
Hsinchu, Taiwan 300, Republic of China

Received 20 July 1992
Revised 10 December 1992

A single-layer recurrent neural network is proposed to perform thinning of binary images. This network iteratively removes the contour points of an object shape by template matching. The set of templates is specially designed for a one-pass parallel thinning algorithm. The proposed neural network produce the same results as the algorithm. Neurons in the neural network performs a sigma-pi function to collect inputs. To obtain this function, the templates used in the algorithm are transformed to equivalent Boolean expressions. After the neural network converges, a perfectly 8-connected skeleton is derived. Good experimental results show the feasibility of the proposed approach.

1. Introduction

Thinning is a process to eliminate redundant pixels from an object shape in an image to obtain the skeleton of the object shape with one-pixel width. Since it is much easier to extract features from a thinned skeleton shape than from a thick one, skeletons obtained by thinning are useful for many image analysis and pattern recognition applications. Algorithms for thinning images can be categorized to two types, namely, sequential and parallel.[1–7] Parallel algorithms differ from sequential ones in the essence that point removal is determined for all the pixels simultaneously. Schemes for sequential thinning cannot be applied to all pixels simultaneously for the reason that it may remove an entire line when the line is thinned to two-pixel width. To prevent this problem, some parallel thinning algorithms divide an iteration into multiple passes.[1,2] On the other hand, one-pass parallel algorithms have also been proposed to save processing time.[3–6] Sequential algorithms are effective when they run on a sequential computer with a single processor. But when a parallel algorithm is simulated on a sequential computer, it is time consuming. So to exert the strength of a parallel algorithm, a multiprocessor system or a special hardware which consists of a large number of simple processing elements is required. For example, Chin *et al.*[5] designed special simple logic gates to match thinning, restoring and trimming templates to enhance thinning speed.

A neural network is a hardware consisting of a large volume of processing elements which cooperates to perform a task in parallel. On the other hand, pixels in an image are usually treated in an identical manner when the image is processed. So it is a good idea to use neural networks to solve image processing problems in parallel. For example, Cortes and Hertz[8] used directional second derivatives to detect the edges in an image by a neural network. Image thinning tasks have also been performed by neural networks in some studies. An example is Matsumoto *et al.*[9] in which images are thinned by cellular neural networks[10,11] with 8 planes. Another example is

* This work was supported partially by National Science Council, Republic of China under grant No. NSC81-0404-E009-017.
† Also with Institute for Information Industry, Taipei, Taiwan 106, Republic of China.
‡ To whom all correspondence should be sent.

Graf *et al.*[12,13] in which a VLSI architecture is designed to multiply a binary input vector with a stored matrix of weights to skeletonize binary images.

A parallel thinning algorithm called OPPTA was proposed previously by the authors.[14] Based on this algorithm, a single-layer recurrent neural network is proposed in this paper to perform thinning of binary images. This network iteratively removes the edge points of an object shape. In each iteration, the removal of a point is determined by the criterion of whether or not the neighbors of that point match any of a set of templates. Both template matching and point removal tasks are performed by the neural network. The neurons in the neural network performs a sigma-pi function to collect inputs. To obtain this function, the templates used in algorithm OPPTA are transformed to equivalent Boolean expressions. Some experimental results obtained from simulating the proposed neural network are shown to assure the correctness of the neural network.

The remainder of this paper is organized as follows. In Sec. 2, the one-pass parallel thinning algorithm OPPTA is reviewed briefly. The transformation of the templates used in OPPTA into a Boolean expression is described in Sec. 3. Section 4 includes the descriptions of the structure of the proposed neural network and the activation function of each neuron in the network. Section 5 gives the experimental results. And finally, concluding remarks are given in Section 6.

2. Review of a Parallel Thinning Algorithm

Templates derived by analysing all the patterns appearing on the border of an object shape are shown in Fig. 1. By matching these templates with a given object shape, the thinning algorithm OPPTA[14] iteratively eliminates the edge points of the object shape layer by layer in parallel. As shown in Fig. 1, there are twelve 3×3 templates (templates (a), (b) and (e) through (n)), one 3×4 template (template (c)) and one 4×3 template (template (d)). The symbols 'c', '0', '1' and 'x' used in these templates denote the currently tested pixel, a white pixel, a black pixel and a don't-care condition, respectively. These symbols follow the conventional notations while the symbol 'y', also appearing in the templates, is a special one. It does not appear singly in a thinning template and at least one of the pixels represented by the set of symbols 'y' should be a white pixel. The details of the OPPTA algorithm are as follows,

```
Algorithm. OPPTA.
Input. a binary image $f^0$.
Output. the image of the thinning result.
Step 1. $i := 0$.
Step 2. flag := false.
Step 3. Check each pixel of $f^i$. If it is a black
        pixel and its neighbors match any of the
        templates (a) through (n), then change
        it to a white pixel and set flag := true.
```

$$\begin{array}{ccc} 1 & 1 & y \\ 1 & c & 0 \\ 1 & 1 & y \end{array} \qquad \begin{array}{ccc} 1 & 1 & 1 \\ 1 & c & 1 \\ y & 0 & y \end{array} \qquad \begin{array}{cccc} y & 1 & 1 & x \\ 0 & c & 1 & 1 \\ y & 1 & 1 & x \end{array} \qquad \begin{array}{ccc} y & 0 & y \\ 1 & c & 1 \\ 1 & 1 & 1 \\ x & 1 & x \end{array}$$

(a) (b) (c) (d)

$$\begin{array}{ccc} x & 0 & 0 \\ 1 & c & 0 \\ x & 1 & x \end{array} \qquad \begin{array}{ccc} x & 1 & 1 \\ 0 & c & 1 \\ 0 & 0 & x \end{array} \qquad \begin{array}{ccc} 0 & 1 & 0 \\ 0 & c & 1 \\ 0 & 0 & 0 \end{array} \qquad \begin{array}{ccc} x & 1 & x \\ 1 & c & 0 \\ x & 0 & 0 \end{array} \qquad \begin{array}{ccc} 0 & 0 & x \\ 0 & c & 1 \\ x & 1 & 1 \end{array} \qquad \begin{array}{ccc} 0 & 0 & 0 \\ 0 & c & 1 \\ 0 & 1 & 0 \end{array}$$

(e) (f) (g) (h) (i) (j)

$$\begin{array}{ccc} 0 & 0 & 0 \\ 0 & c & 0 \\ 1 & 1 & 1 \end{array} \qquad \begin{array}{ccc} 1 & 0 & 0 \\ 1 & c & 0 \\ 1 & 0 & 0 \end{array} \qquad \begin{array}{ccc} 1 & 1 & 1 \\ 0 & c & 0 \\ 0 & 0 & 0 \end{array} \qquad \begin{array}{ccc} 0 & 0 & 1 \\ 0 & c & 1 \\ 0 & 0 & 1 \end{array}$$

(k) (l) (m) (n)

Fig. 1. Thinning templates of algorithm OPPTA.

Step 4. If flag = false, which means the image is thinned, then go to Step 5 with f^i as the thinning result. Otherwise, $i := i + 1$ and go to Step 2 to perform the next iteration.

Step 5. Output the thinning result.

As discussed in the previous paper,[14] OPPTA obtains good thinning results and is guaranteed to terminate after a proper number of iterations. Briefly, we summarize the characteristics of OPPTA in the following.

(1) It preserves the connectivity of an object shape.
(2) It prevents excessive erosion.
(3) It is noise insensitive.
(4) It produces skeletons topologically equivalent to original object shapes.

3. Equivalent Boolean Expression

The principle of designing the proposed recurrent neural network is as follows. First, each neuron in the proposed neural network is allowed the ability to perform a sigma-pi function on the inputs of the neuron. Next, without considering the negation of a literal, a Boolean expression in disjunctive normal form is exactly in the form of a sigma-pi function. Furthermore, a template matching scheme can be transformed to an equivalent Boolean expression so that at least one of the templates is matched if and only if the Boolean expression is evaluated to be true. After the equivalent Boolean expression is derived, the sigma-pi expression can be derived. In the remainder of this section, the transformation of the templates into an equivalent Boolean expression will be discussed.

First, consider a template t with a pair of symbols 'y' and recall that at least one of this pair should be a white pixel. We can generate two new templates $t1$ and $t2$ from t by substituting one symbol of 'y' in template t with symbol '0' and the other with symbol 'x' to obtain template $t1$; and substituting the two symbols of 'y' in template t with symbols 'x' and '0' in a reverse order to obtain template $t2$. For example, template (a) can be expanded to templates (a1) and (a2) as shown in Fig. 2. Templates (b) through (d) can be expanded similarly with the results shown in Fig. 2. In the following discussions, we will use the new template set consisting of templates (a1) through (d2) in Fig. 2 as well as templates (e) through (n) in Fig. 1. Symbol 'y' is therefore excluded from the templates. The above process is necessary before deriving equivalent Boolean expressions from the templates.

As described previously, the purpose of deriving an equivalent Boolean expression is to get the corresponding sigma-pi function. The literals appearing in the derived equivalent Boolean expression can be considered to be corresponding to the involved pixels when the templates are matched. In order to derive the literals of the Boolean expression, we have to define the pixels which should be considered when a template is matched. If all of the templates were in size of 3 × 3, the area which should be matched against when a pixel p is tested in the thinning process is the 8-neighborhood of p. However, the addition of the 3 × 4 templates and the 4 × 3 templates (i.e. templates (c1) through (d2) in Fig. 2) increases

```
1 1 0        1 1 x        1 1 1      1 1 1
1 c 0        1 c 0        1 c 1      1 c 1
1 1 x        1 1 0        0 0 x      x 0 0

 (a1)         (a2)         (b1)       (b2)

0 1 1 x      x 1 1 x      0 0 x      x 0 0
0 c 1 1      0 c 1 1      1 c 1      1 c 1
x 1 1 x      0 1 1 x      1 1 1      1 1 1
                          x 1 x      x 1 x
 (c1)         (c2)         (d1)       (d2)
```

Fig. 2. Templates without symbol 'y'.

the number of neighbors which should be considered from 8 to 14. Observing templates (c1) through (d2), we see that except the 8-neighbors in a 3×3 window, each template of (c1) through (d2) includes further only one important neighbor (denoted as '1') since the other two neighbors (denoted as 'x') are don't-cares. We can thus define a 10-neighborhood for each tested pixel. As shown in Fig. 3(a), any of the set of pixels appearing in the figure around the tested pixel P_{ij} is defined as a 10-neighbor of P_{ij}. In addition to the 8-neighbors in a 3×3 window, the east-neighbor of the east-neighbor of P_{ij} (i.e. $P_{i,j+2}$) and the south-neighbor of the south-neighbor of P_{ij} (i.e. $P_{i+2,j}$) are also included. This reduces the number of concerned neighbors from 14 to 10. For a better representation of the Boolean expression, new labels for these neighbors are given in Fig. 3(b). These labels will also be used in the literals appearing in the derived Boolean expression.

Let each black pixel and white pixel in the image correspond to the logical values of TRUE and FALSE, respectively, when a Boolean expression is evaluated. A template then can be transformed to a product clause in the following way;

(1) For all positions specified by symbol '1' in the template, add their corresponding literals to the product clause.
(2) For all positions specified by symbol '0' in the template, add their corresponding negative literals to the product clause.

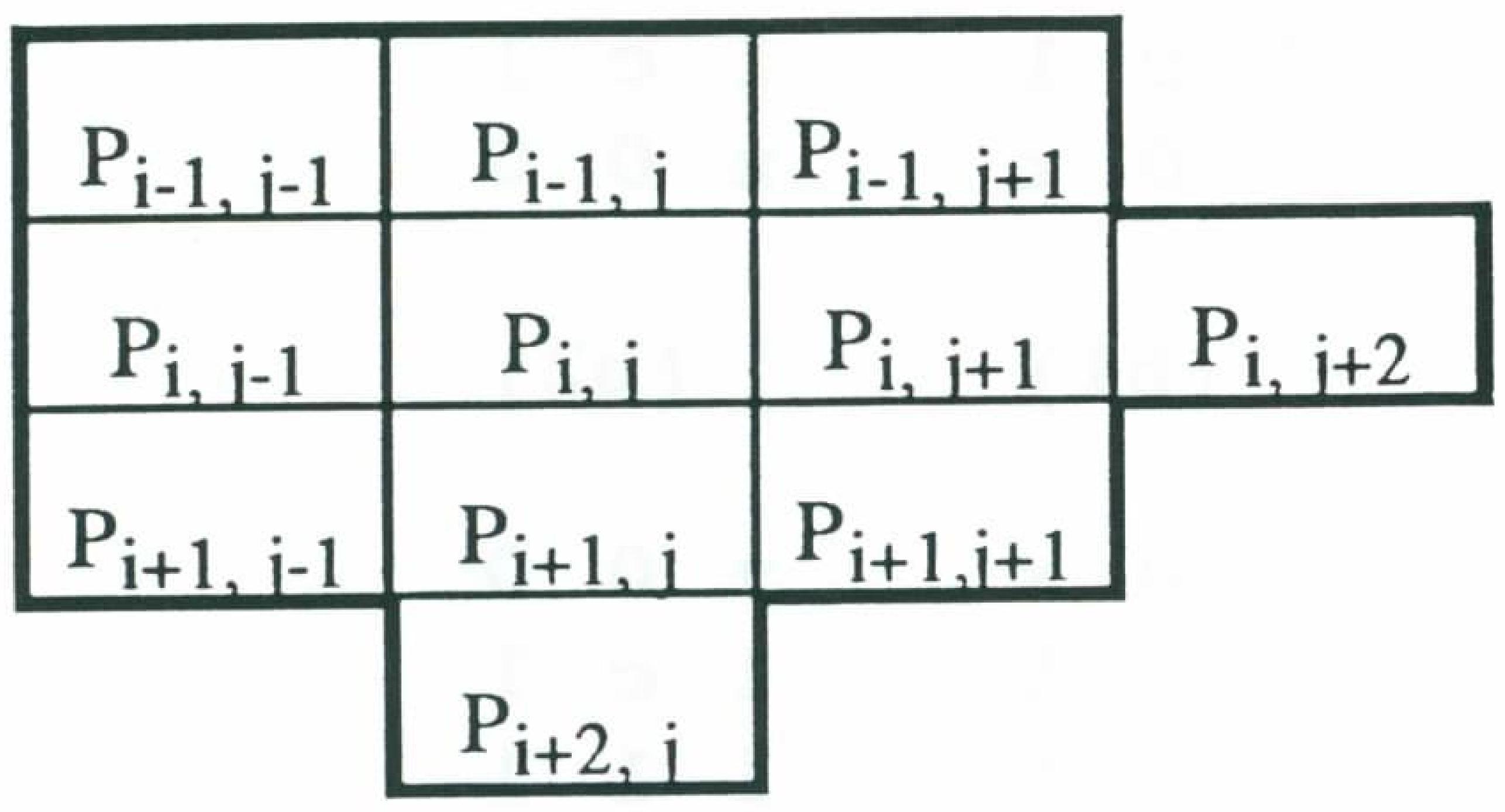
For example, if each logical negation is represented by an apostrophe, then template (a1) is transformed into

a1 : $X_1X_2X_3'X_4'X_6X_7X_8$.

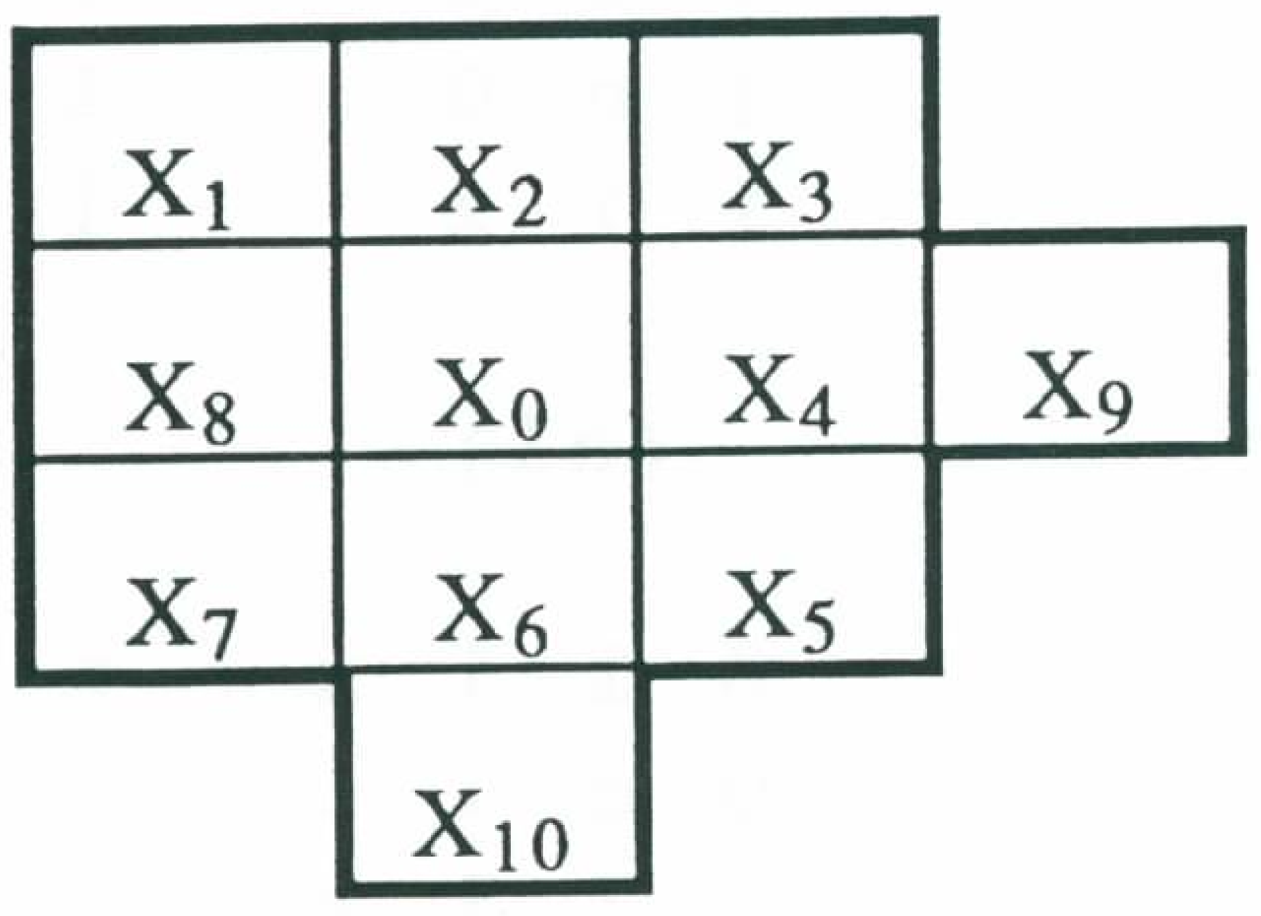
Similarly, templates (a2) through (n) can be transformed into

a2 : $X_1X_2X_4'X_5'X_6X_7X_8$
b1 : $X_1X_2X_3X_4X_6'X_7'X_8$
b2 : $X_1X_2X_3X_4X_5'X_6'X_8$
c1 : $X_1'X_2X_3X_4X_5X_6X_8'X_9$
c2 : $X_2X_3X_4X_5X_6X_7'X_8'X_9$
d1 : $X_1'X_2'X_4X_5X_6X_7X_8X_{10}$
d2 : $X_2'X_3'X_4X_5X_6X_7X_8X_{10}$
e : $X_2'X_3'X_4'X_6X_8$
f : $X_2X_3X_4X_6'X_7'X_8'$
g : $X_1'X_2X_3'X_4X_5'X_6'X_7'X_8'$
h : $X_2X_4'X_5'X_6'X_8$
i : $X_1'X_2'X_4X_5X_6X_8'$
j : $X_1'X_2'X_3'X_4X_5'X_6X_7'X_8'$
k : $X_1'X_2'X_3'X_4'X_5X_6X_7X_8'$
l : $X_1X_2'X_3'X_4'X_5'X_6'X_7X_8$
m : $X_1X_2X_3X_4'X_5'X_6'X_7'X_8'$
n : $X_1'X_2'X_3X_4X_5X_6'X_7'X_8'$.

Since any pixel which does not match the corresponding pixel in a template contributes a value of FALSE to the transformed Boolean product clause, the determination of whether a template is matched or not is equivalent to the evaluation of the logical value of the transformed product clause. By ORing all product clauses, a Boolean expression E is derived for the entire thinning process:



(a)



(b)

Fig. 3. 10-neighbors of pixel P_{ij}.

$$
\begin{aligned}
E = {} & X_1X_2X_3'X_4'X_6X_7X_8 \\
& + X_1X_2X_4'X_5'X_6X_7X_8 \\
& + X_1X_2X_3X_4X_6'X_7'X_8 \\
& + X_1X_2X_3X_4X_5'X_6'X_8 \\
& + X_1'X_2X_3X_4X_5X_6X_8'X_9 \\
& + X_2X_3X_4X_5X_6X_7'X_8'X_9 \\
& + X_1'X_2'X_4X_5X_6X_7X_8X_{10} \\
& + X_2'X_3'X_4X_5X_6X_7X_8X_{10} \\
& + X_2'X_3'X_4'X_6X_8 \\
& + X_2X_3X_4X_6'X_7'X_8' \\
& + X_1'X_2X_3'X_4X_5'X_6'X_7'X_8' \\
& + X_2X_4'X_5'X_6'X_8 \\
& + X_1'X_2'X_4X_5X_6X_8' \\
& + X_1'X_2'X_3'X_4X_5'X_6X_7'X_8' \\
& + X_1'X_2'X_3'X_4'X_5X_6X_7X_8' \\
& + X_1X_2'X_3'X_4'X_5'X_6'X_7X_8 \\
& + X_1X_2X_3X_4'X_5'X_6'X_7'X_8' \\
& + X_1'X_2'X_3X_4X_5X_6'X_7'X_8' .
\end{aligned}
$$

Since the determination of whether a certain template is matched or not is equivalent to the evaluation of its corresponding product clause, the evaluation of the logical value of Boolean expression E is equivalent to the determination of whether there is a template among those of Figs. 1 and 2 which matches the input pattern.

4. Proposed Neural Network

The architecture of the proposed neural network is in the form of a single neuron plane. The neurons in this plane and those pixels in the input image have one-to-one correspondences. As shown in Fig. 4, each neuron receives inputs from its 10-neighbors and itself. Initially, the input image is fed into this plane by activating those neurons which correspond to the black pixels and leaving the other neurons inactive. After the network becomes stable, the image of the thinning result will appear in this plane with active neurons representing black pixels. When performing the thinning work, this plane represents the temporary thinning result.

Expression E obtained in the previous section is a Boolean expression. On the other hand, the sigma-pi function performed by the neurons is an arithmetic operation. So transforming the Boolean expression E to a corresponding arithmetic expression is required for the operation of a neuron. The

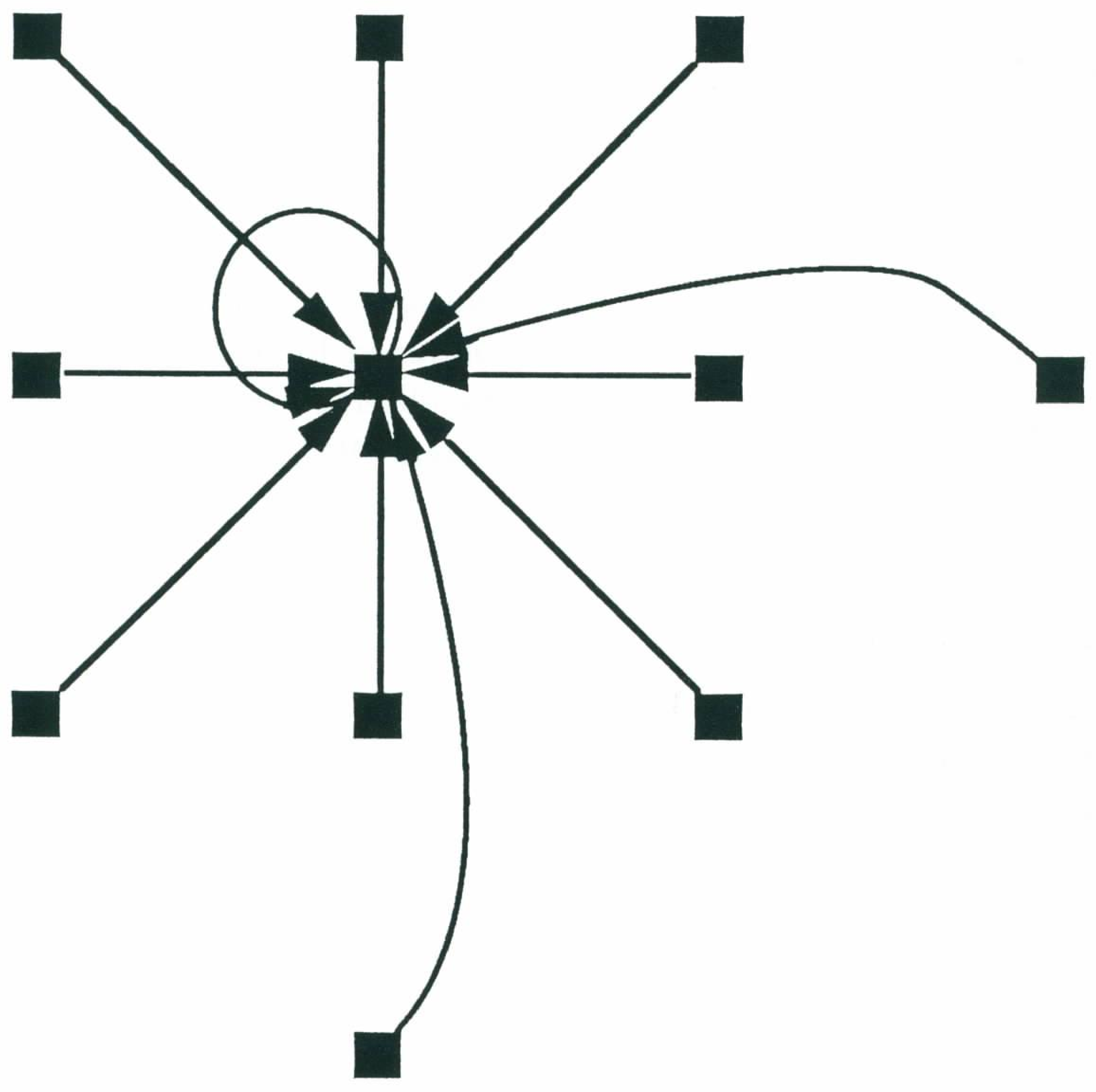


Fig. 4. Connections between neurons.

transformation can be performed simply by treating the literals in expression E as real variables, the Boolean AND operation as the arithmetic product and the Boolean OR operation as the arithmetic addition and changing a negative literal X_i' to $(1 - X_i)$. An arithmetic expression S can then be derived to be

$$\begin{aligned}
S = {} & X_1X_2(1-X_3)(1-X_4)X_6X_7X_8 \\
& + X_1X_2(1-X_4)(1-X_5)X_6X_7X_8 \\
& + X_1X_2X_3X_4(1-X_6)(1-X_7)X_8 \\
& + X_1X_2X_3X_4(1-X_5)(1-X_6)X_8 \\
& + (1-X_1)X_2X_3X_4X_5X_6(1-X_8)X_9 \\
& + X_2X_3X_4X_5X_6(1-X_7)(1-X_8)X_9 \\
& + (1-X_1)(1-X_2)X_4X_5X_6X_7X_8X_{10} \\
& + (1-X_2)(1-X_3)X_4X_5X_6X_7X_8X_{10} \\
& + (1-X_2)(1-X_3)(1-X_4)X_6X_8 \\
& + X_2X_3X_4(1-X_6)(1-X_7)(1-X_8) \\
& + (1-X_1)X_2(1-X_3)X_4(1-X_5) \\
& \quad \times (1-X_6)(1-X_7)(1-X_8) \\
& + X_2(1-X_4)(1-X_5)(1-X_6)X_8 \\
& + (1-X_1)(1-X_2)X_4X_5X_6(1-X_8) \\
& + (1-X_1)(1-X_2)(1-X_3)X_4(1-X_5) \\
& \quad \times X_6(1-X_7)(1-X_8) \\
& + (1-X_1)(1-X_2)(1-X_3) \\
& \quad \times (1-X_4)X_5X_6X_7(1-X_8) \\
& + X_1(1-X_2)(1-X_3)(1-X_4) \\
& \quad \times (1-X_5)(1-X_6)X_7X_8 \\
& + X_1X_2X_3(1-X_4)(1-X_5) \\
& \quad \times (1-X_6)(1-X_7)(1-X_8) \\
& + (1-X_1)(1-X_2)X_3X_4X_5(1-X_6) \\
& \quad \times (1-X_7)(1-X_8)\,.
\end{aligned}$$

A good relationship existing between a real variable in expression S and a Boolean literal in expression E is that a real number of 0 or 1 corresponds to a Boolean value of FALSE or TRUE, respectively. That is, arithmetic expression S will be evaluated to produce a value greater than or equal to 1 if and only if Boolean expression E is evaluated to be TRUE. By defining the active and inactive values of a neuron to be 1 and 0, respectively, the net input of each neuron can be defined to be

$$N = W_1 * X_0 + W_2 * S\,,$$

where X_0 is the output of the neuron itself in the previous iteration and W_1 and W_2 are connection weights with the values of 1 and -1, respectively. The net input N of a neuron will be greater than or equal to 1 if and only if X_0 is equal to 1 and S is equal to 0 (which means that there is no template being matched). Considering the neuron corresponding to a border point, the net input N will be less than or equal to 0 since S is greater than or equal to 1.

Finally, define the neuron output as

$$X_0 = F(N)\,,$$

where the activation function $F(x)$ is defined as

$$F(x) = \begin{cases} 1 & \text{if } x \geq 1 \\ 0 & \text{otherwise.} \end{cases}$$

This activation function will activate a neuron if and only if the neuron is active in the previous iteration and S is equal to 0. So the border points of an object image will be removed layer by layer by the proposed neural network. Since algorithm OPPTA is guaranteed to halt, the proposed neural network is also guaranteed to converge after a number of iterations.

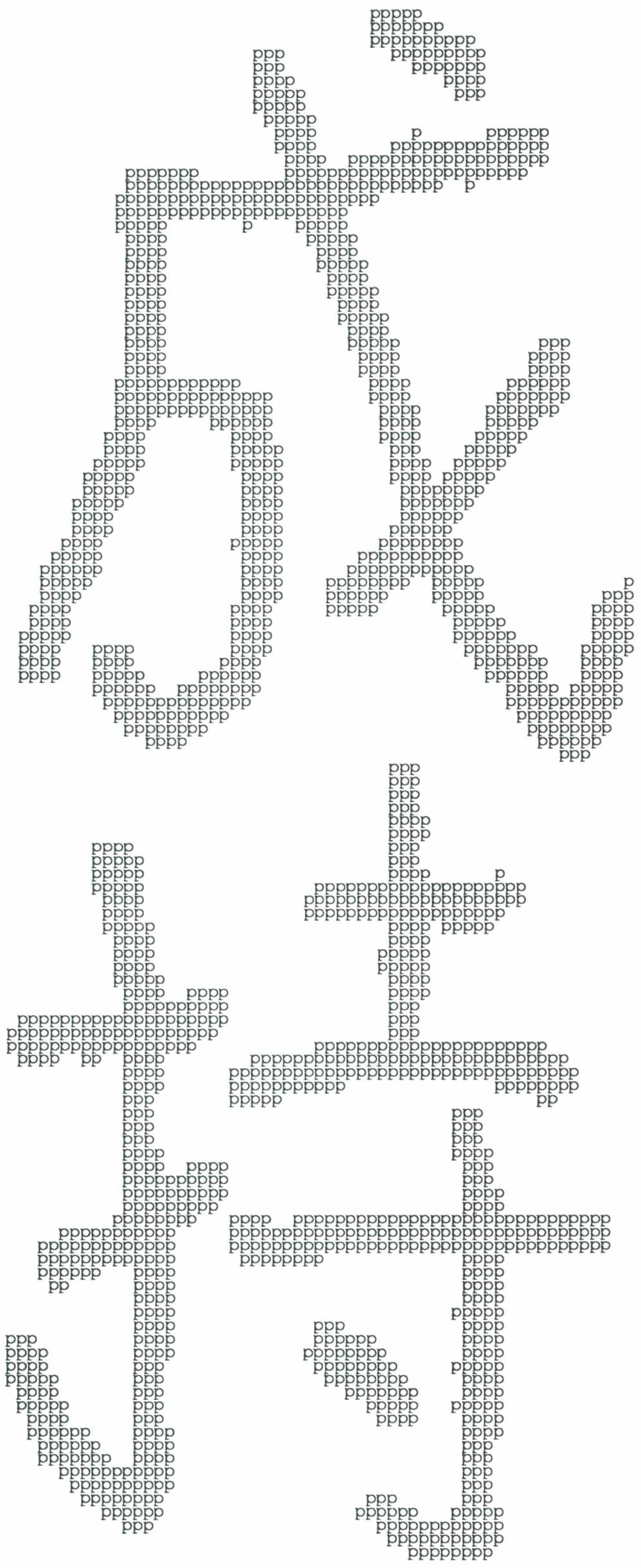
5. Experimental Results

Some experiments have been conducted to illustrate the effects of the proposed neural network. Figures 5 and 6 show the results in which skeleton points are marked by 'p' and removed points by '.'.

Figure 5 shows the results of thinning four Chinese characters in dimension 64 × 64. It can be seen that the proposed neural network obtains perfectly 8-connected and noise-insensitive skeletons without excessive erosion, exactly the same as those obtained by algorithm OPPTA. Similar results can be observed in the thinning results of English characters 'B' and 'H' in dimension 64 × 64 shown in Fig. 6.

6. Conclusions

A one-layer neural network for thinning binary images has been proposed. It produces the same results as the parallel algorithm OPPTA proposed previously for thinning. The proposed neural network is simple. This comes from two facts. The first is the simplification of the symbols in the templates (i.e. removal of 'y') which makes possible the derivation of an equivalent Boolean expression. The second is to allow each neuron to collect all its inputs by the use of a sigma-pi function. It is noted finally


(a) Original image.

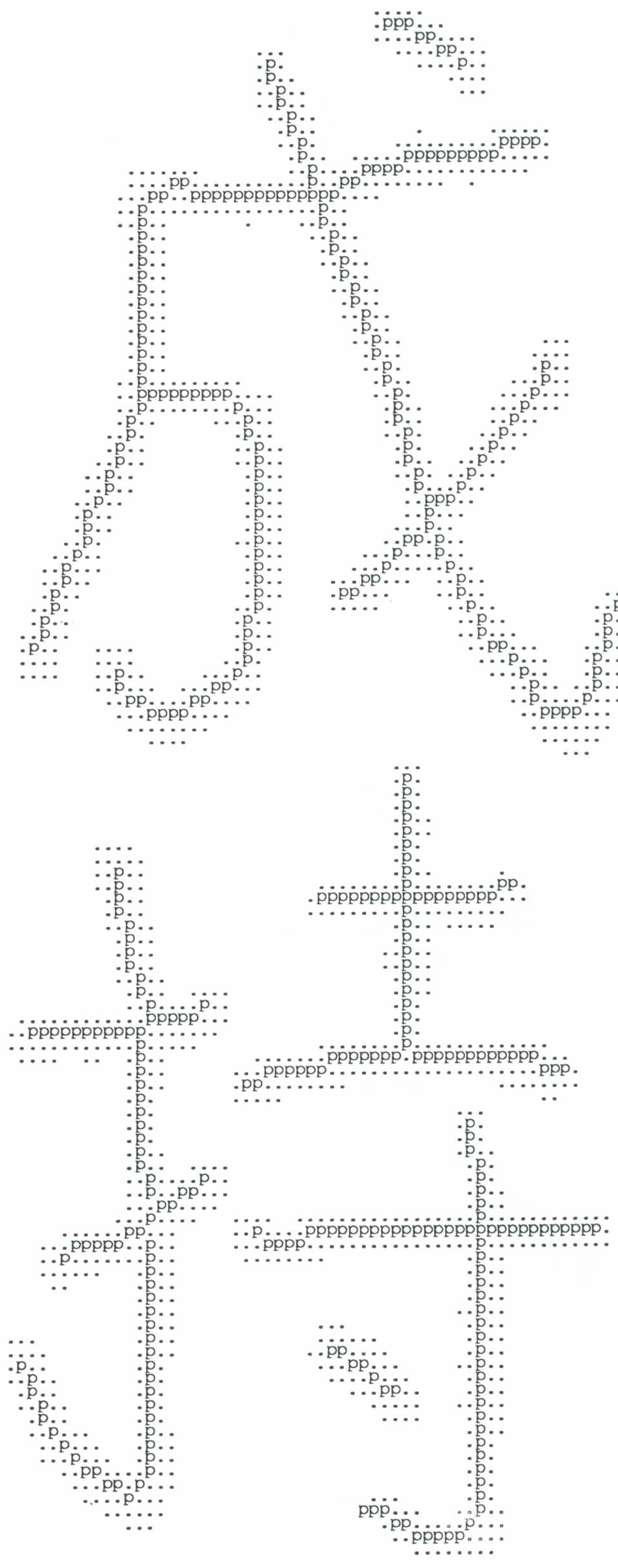
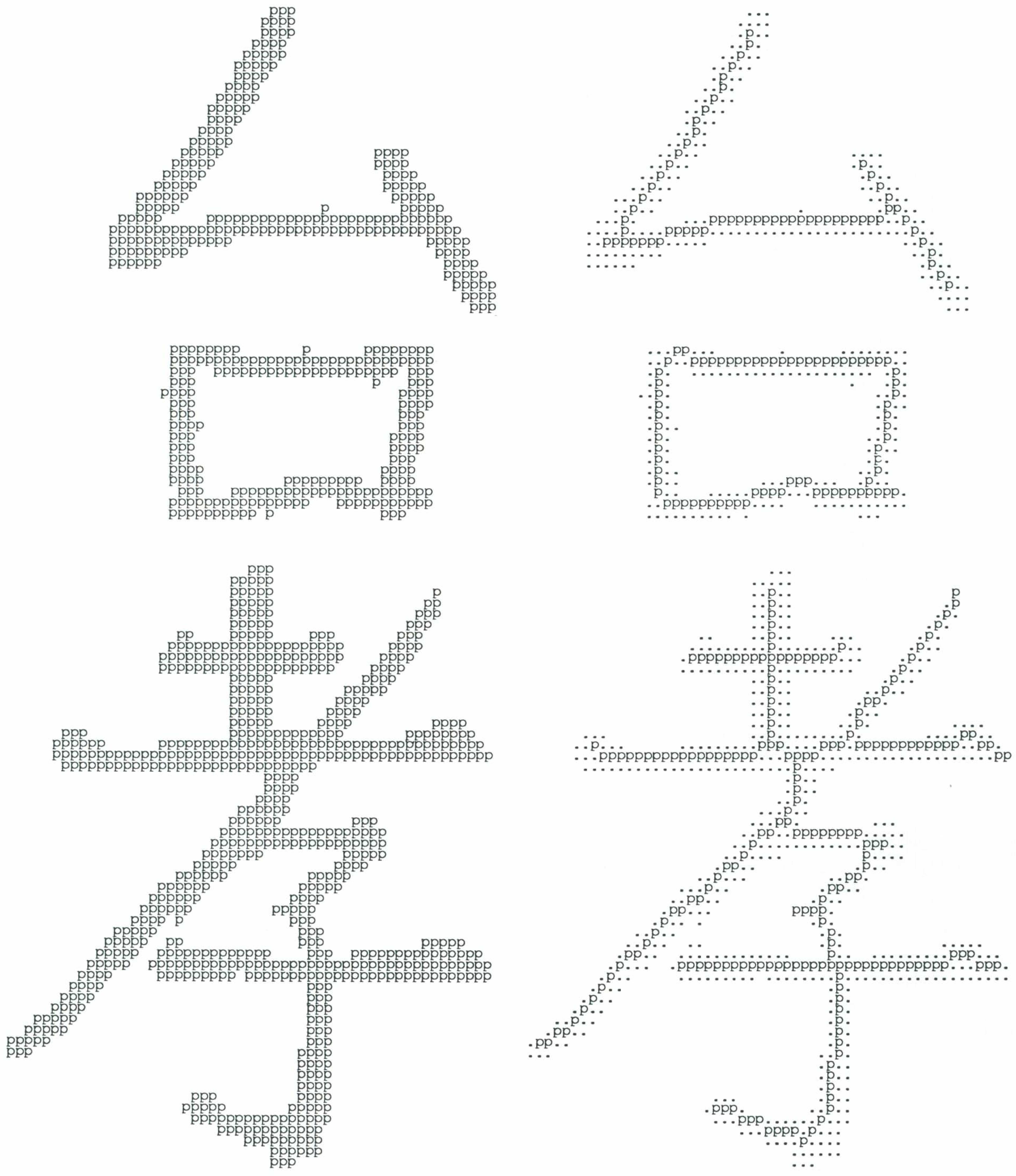
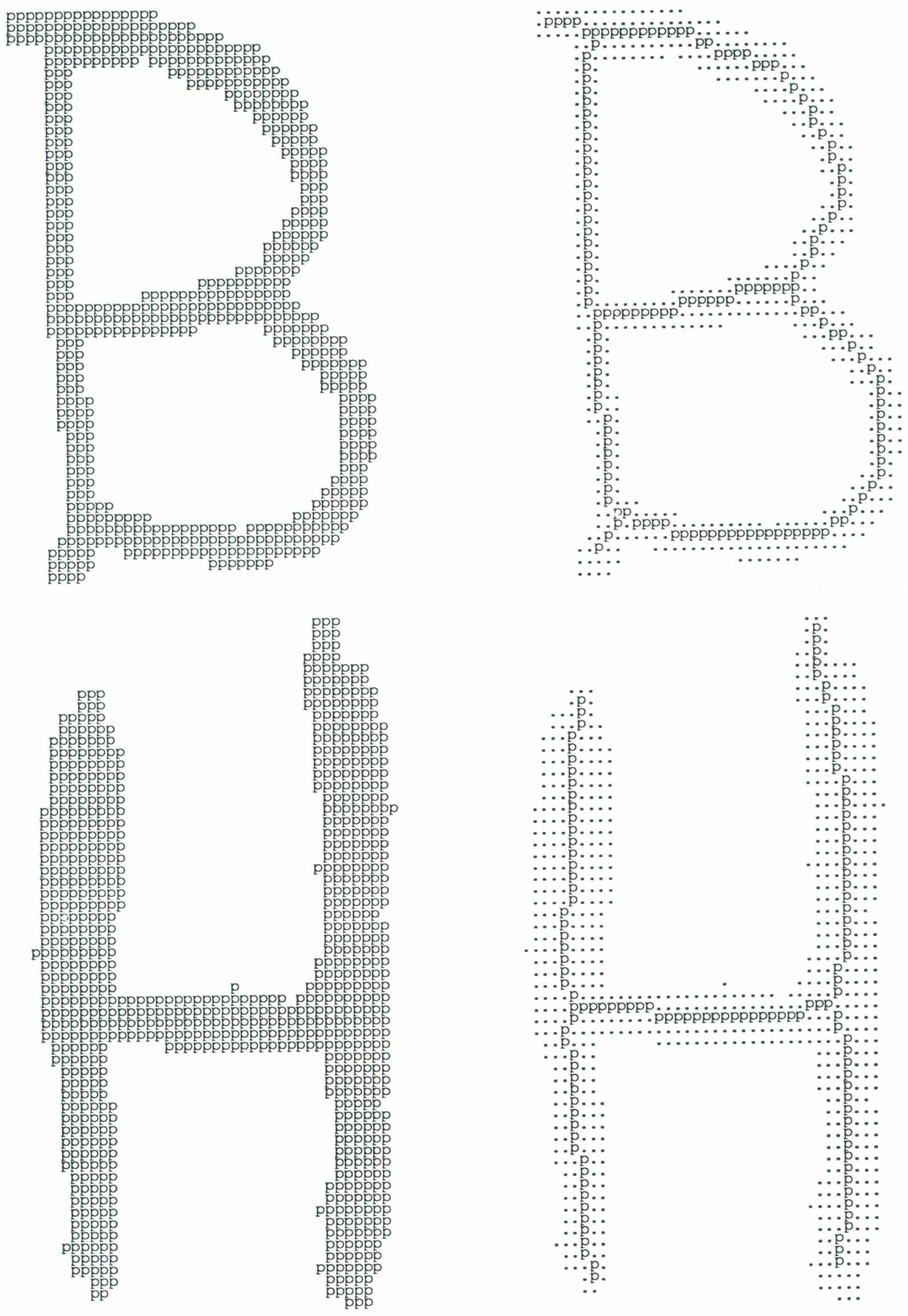

(b) Thinning result.

Fig. 5. Thinning results of four Chinese characters by proposed neural network.



(a) Original image. (b) Thinning result.

Fig. 5. (*Continued*).



(a) Original image. (b) Thinning result.

Fig. 6. Thinning result of English characters 'B' and 'H' by proposed neural network.

that the proposed method of transforming the parallel thinning algorithm OPPTA into the single-layer neural network actually is a general scheme and can be applied to other parallel thinning algorithms.

References

1. R. Stefanelli and A. Rosenfeld, "Some parallel thinning algorithms for digital pictures," *J. ACM* **18:2**, 255–264 (1971).
2. T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM* **27:3**, 236–239 (1984).
3. Z. Guo and R. W. Hall, "Parallel thinning with two-subiteration algorithms," *Commun. ACM* **32:3**, 359–373 (1989).
4. C. M. Holt, A. Stewart, M. Clint and R. H. Perrott, "An improved parallel thinning algorithm," *Commun. ACM* **30:2**, 156–160 (1987).
5. R. T. Chin, H. K. Wan, D. L. Stover and R. D. Iverson, "A one-pass thinning algorithm and its parallel implementation," *Computer Vision, Graphics, and Image Processing* **40**, 30–40 (1987).
6. C. S. Chen and W. H. Tsai, "A new fast one-pass thinning algorithm and its parallel hardware implementation," *Pattern Recog. Lett.* **11**, 471–477 (1990).
7. S. S. Yu and W. H. Tsai, "A new thinning algorithm for gray-scale images by the relaxation technique," *Pattern Recog.* **23:10**, 1067–1076 (1990).
8. C. Cortes and J. A. Hertz, "A network system for image segmentation," in *Proc. Int. Joint Conf. Neural Networks* 121–125 (1989).
9. T. Matsumoto, L. O. Chua, and T. Yokohama, "Image thinning with a cellular neural network," *IEEE Trans. Circuits and Syst.* **37:5**, 638–640 (1990).
10. L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits and Syst.* **35:10**, 1257–1272 (1988).
11. L. O. Chua and L. Yang, "Cellar Neural Networks: Application," *IEEE Trans. Circuits and Syst.* **35:10**, 1273–1290 (1988).
12. H. S. Baird, H. P. Graf, L. D. Jackel and W. E. Hubbard, "A VLSI architecture for binary image classification," in *From Pixels to Features*, ed. J. C. Simon (North-Holland, Amsterdam, 1989).
13. H. P. Graf, L. D. Jackel and W. E. Hubbard, "VLSI implementation of a neural network model," *IEEE Computer* 41–48 (1988).
14. R. Y. Wu and W. H. Tsai, "A new one-pass parallel thinning algorithm for binary images," *Pattern Recog. Lett.* **13**, 715–723 (1992).