5-1990

Korean Character Recognition Using Neural Networks

Jinhwan Koh
Syracuse University

G. S. Moon

Kishan Mehrotra
Syracuse University, mehrotra@syr.edu

Chilukuri K. Mohan
Syracuse University, ckmohan@syr.edu

Sanjay Ranka
Syracuse University

SU-CIS-90-06

Korean Character Recognition Using Neural Networks

J. Koh, G.S. Moon, K. Mehrotra, C. K. Mohan, and S. Ranka

May 1990

School of Computer and Information Science
Syracuse University
Suite 4-116
Center for Science and Technology
Syracuse, New York 13244-4100

Korean Character Recognition using Neural Networks

J. Koh, G. S. Moon, K. Mehrotra, C. K. Mohan, and S. Ranka
School of Computer and Information Science
Syracuse University, NY 13244-4100

Abstract

We present a neural-network approach for recognizing printed Korean characters. Our approach is based on a variant of the back-propagation algorithm. The results indicate that by transforming the character data into Hough space, we can achieve excellent recognition.

1.0 Introduction

Considerable research has been accomplished on character recognition for the Roman script. Korean character recognition is a much more complex task, since each 'character' is itself a complex combinations of strokes representing simple components or sub-characters.

There are 24 basic components in Korean characters (10 vowels and 14 consonants). These basic components are used to construct structures called doubled components and combined components (Figure 1).

basic components

vowels : ㅏ ㅑ ㅓ ㅕ ㅗ ㅛ ㅜ ㅠ ㅡ ㅣ

consonants : ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ

doubled components

consonants : ㄲ ㄸ ㅃ ㅆ ㅉ

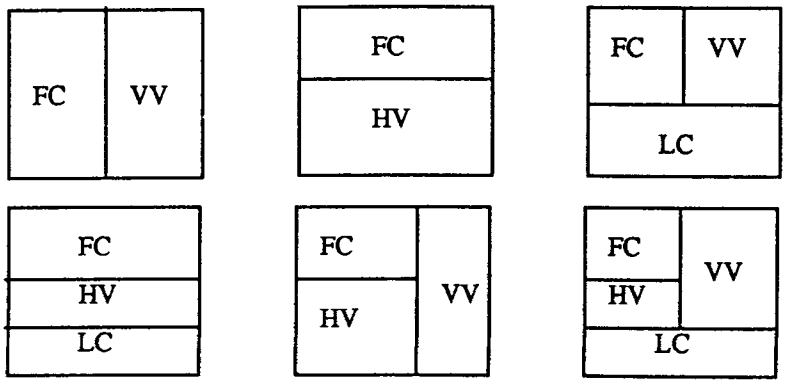
combined components

vowels : ㅐ ㅒ ㅔ ㅖ ㅚ ㅟ ㅢ ㅙ ㅞ ㅘ ㅝ

consonants : ㄳ ㄵ ㄶ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅄ

Figure 1

Each Korean character is a complex combination of 2 to 6 components, which are put together in phonetic modules (FC,LC,VV,HV) in a rectangular arrangement. All character compositions belong to one of six composition forms, shown in Figure 2.



FC : first consonant ; basic components + doubled components

LC : last consonant ; basic components + combined components + (ㄲ ㅆ)

HV : horizontal vowel ; basic components + combined components

VV : vertical vowel ; basic components + combined components

Figure 2 : Six different methods of combining components into Korean characters

Thus there are 19x21x27 = 10,773 different possible characters. This large number of characters makes recognition of Korean characters extremely difficult. In addition, each phonetic component can have several shapes, depending upon its location in a character. For instance, the consonant "gierk" (ㄱ) has about 16 different shapes corresponding to its different locations. Many techniques for automatic Korean character recognition have been described in [1], [2], and [3]. These techniques are based on the extraction of five basic letter components (| ㅡ / \ ○). In this paper we present an approach for recognizing Korean characters using neural networks. We transform the input character image into Hough space such that lines in the original image map into points in Hough space. The neural network for recognizing Korean character is applied to each phonetic module (FC, LC, VV, HV) segmented from a character. Our results show that we can recognize FC components with a performance of up to 62% for the training samples and 51% for the testing samples, and also recognize LC components up to 88% for the training samples and 85% for the testing samples, and VV components up to 100% for the training samples and 88% for the testing samples, and last HV components up to 100% for the training samples and 87% for the testing samples respectively. The average recognition for all modules is thus 81% for training samples and 73% for testing samples.

The rest of the paper is structured as follows. Section 2 gives a brief explanation of thinning, segmentation of a character into each category component (FC, LC, VV & HV), and feature extraction in Hough space. Section 3 describes our neural network architecture. In Section 4, we give details of our training and testing results, and discuss these.

2.0 Preprocessing of character images

The raw data, consisting of a pixel array, is difficult to process using a neural network, due to the large number of neurons and connections required, and consequently the prohibitively large training time. In our approach, we hence preprocess the image, extracting a description of each character in terms of the line segments it contains; the neural network receives such a description as input. The basic preprocessing strategy is given in Figure 3.

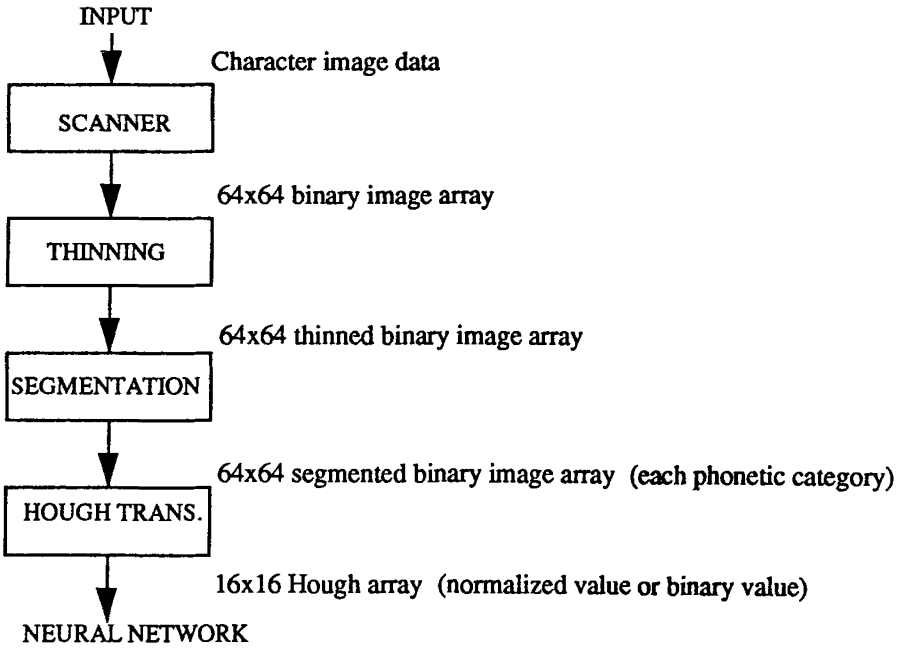


Figure 3 : Basic strategy of preprocessing character images

The character image data goes through the digital image scanner and a 64x64 binary image array is obtained for each character. A thinning technique [4] is applied to this binary image array in order to get feature points which are used in segmenting a character into each phonetic module (FC,LC,VV,and HV).

The last stage in preprocessing is to apply Hough transform to each phonetic module. The original Hough transform is a method for line detection in images [5]; it has been generalized to detect shapes other than straight lines [6]. All components of Korean characters are composed of circles and straight line segments, which can be extracted in Hough space. There are a few different parameterizations of Hough transform [7]. We use the angle of normal (θ), and the distance to the line from the origin (ρ) as parameters of Hough transform. The algorithm for calculating the Hough array $H(\theta,\rho)$ is as follows;

```
For each (x,y) do
    For θ from 0 to 2π in increments of Δθ do
        ρ = x cos θ + y sin θ
        H(θ,ρ) = H(θ,ρ) + 1 (if ρ > 0) or (if ρ = 0 and 0 ≤ θ ≤ π)
    end
end
```

The quantity $\Delta\theta$ defines the sampling of θ. The efficiency of the algorithm is due to the fact that ρ is determined completely by x,y,and θ. The size of Hough transform was chosen to the 16x16 to keep the size of neural network reasonable. This size was tuned by experimentation and found to be efficient in performing excellent recognition.

3.0 Neural Network Architecture

Following a modular design technique, the network is divided into 4 parts, each part being designed to analyze one phonetic category (FC, LC, VV, HV). For convenience, we refer to these parts as the FC module, LC module, VV module, and HV module, respectively. Every module has 16 x 16 input units and 8 hidden units. The number of output units of each module is same as the number of phonetic components included in that module. That is, FC module has 19 output units, LC module has 21 output units, and each of VV module and HV module has 5 output units. In the vowel module, we consider just 10 vowels, which are single components, because of a limitation of segmentation algorithm used. Namely, VV module treats vowels "a" (ㅏ), "er" (ㅓ), "ie" (ㅣ), "ya" (ㅑ),"yer" (ㅕ), and HV module treats vowels "ee" (ㅡ), "o" (ㅗ), "oo" (ㅜ), "yo" (ㅛ), "you" (ㅠ). The MAXNET is connected to the output of each module. The maximum output, therefore, is selected as the output of that module. The overall network diagram is shown in Figure 4. To help understand the network structure, we show the structure of one module (the VV module) in Figure 5.

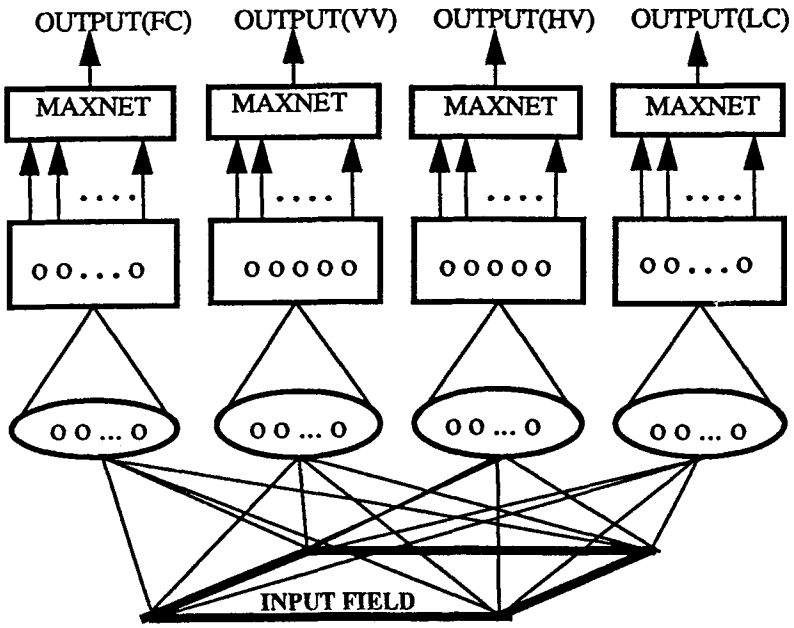


Figure 4 : Neural Network Architecture

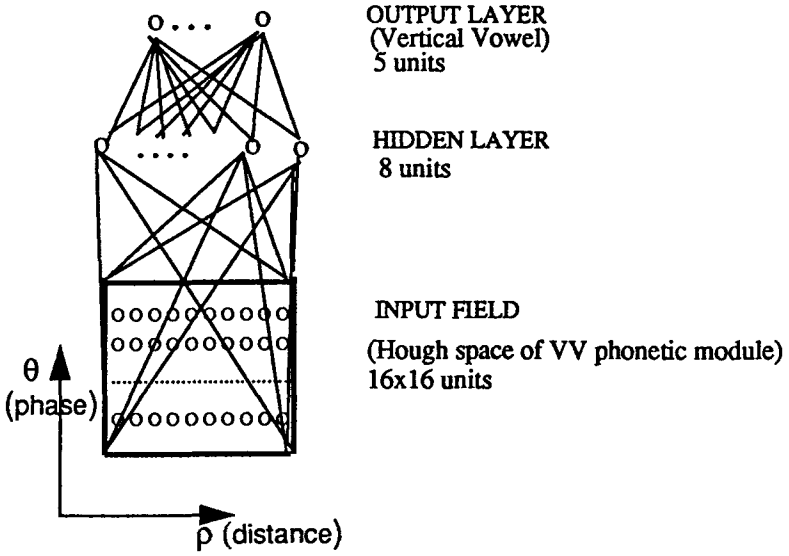


Figure 5 : Network structure of VV module

Each hidden unit is connected to each input unit and each output unit in every module. There are no direct connections between the input units and output units. Input units can operate in one of two possible modes. If input data is binary, the input unit is clamped to the input value. If input data is discrete, the input unit is graded by the input value. The activation function for the output of each unit is a sigmoid (continuous increasing) function and the activation value of each unit is a real value between zero and one.

Each module gets the input from its own input field in Hough space. For example, the input units of FC module are connected to the FC category of Hough space (domain). The input

units of other modules are also connected to the respective category of Hough space. The overall schematic diagram of this system is presented in Figure 6.

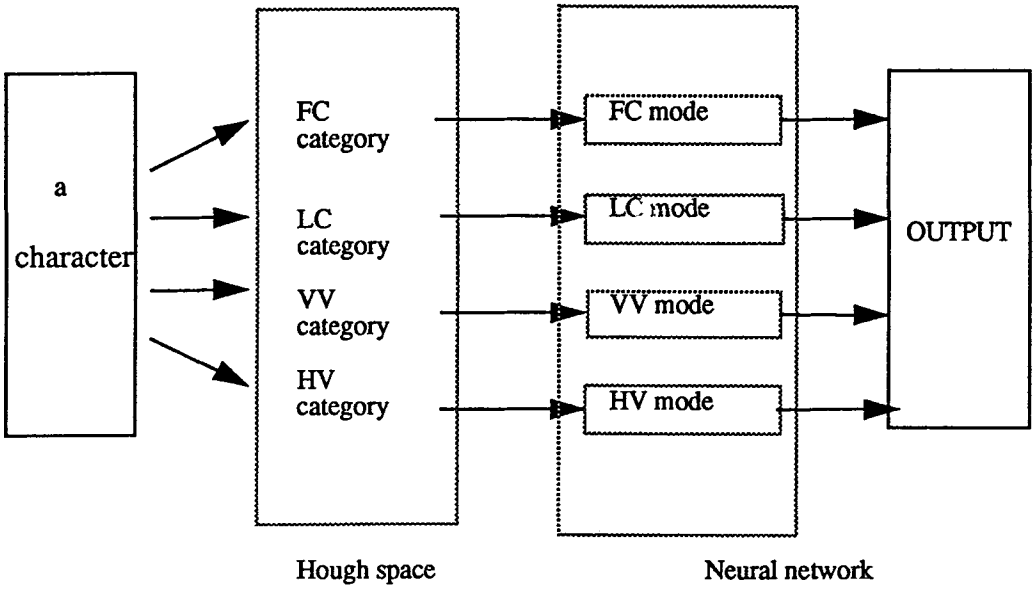


Figure 6 : Overall schematic diagram of recognition system

4.0 Training and Testing

We used the back propagation algorithm to train the network. This well-known learning algorithm based on gradient-descent is described in [8]. The learning procedure is accomplished by the following rule which indicates how the weight of each connection is modified.

$$\Delta w_{ji}(n+1) = \eta (\delta_{pj} O_{pj}) + \alpha \Delta w_{ji} (n)$$

Notation : η is the learning rate, α is the momentum constant for rapid learning, n is the time index, w_{ji} is the weight from unit i to unit j, p is an index over the cases, and δ_{pj} is the propagated error signal seen at unit j in case p, and O_{pj} is the output of the corresponding unit. For the sigmoid activation function, where

$$O_{pj} = \frac{1}{1 + \exp(-\sum_{i} w_{ji} O_{pi} - \theta_j)} ,$$

the error signal is given by

$$\delta_{pj} = (t_{pj} - O_{pj}) O_{pj} (1 - O_{pj}) , \quad \text{for an output unit,}$$

$$\delta_{pj} = O_{pj}(1 - O_{pj}) \sum_{k} \delta_{pk} w_{kj} , \quad \text{for a hidden unit,}$$

where t_{pj} is the teaching input for output unit j of the case p.

In our work, we used 0.5 as the value for the momentum constant α. The learning rate was varied from 0.5 to 0.2 after 100 cycles ('epochs'). All patterns are processed once in a cycle. For the teaching input, we used some bias teaching values for the incorrect output unit to enforce a little learning in case of no teaching input, i.e., 1.0 is used for the teaching input to the correct output unit and 0.25 (rather than zero) is used for the teaching input to the other output units.

For testing the performance of this network in recognizing Korean characters, we calculated success rate in training mode by counting the frequency samples which are correctly classified at a given training cycle.

$$\text{success rate} = \frac{\text{no. of training samples correctly classified}}{\text{total no. of training samples}} \quad (\times 100\ \%)$$

Another performance measure is obtained by testing the network on new character samples which were <u>not</u> used to train the network. This is obtained as a ratio of the number of testing samples correctly classified to the total number of testing samples.

4.1 Input Data (character set used in this paper)

No standard Korean character data base was available. So we used a Korean character set in [4] in which there are 50 characters in each combination category (FC-VV combination, FC-HV combination, and so on). We performed character recognition on 4 combination categories (FC-VV, FC-HV, FC-VV-LC, and FC-HV-LC) because of a limitation of segmentation algorithm used. There were 200 different characters used. The respective number of each phonetic component used in this paper is listed in the test results in section 4.2.

4.2 Testing

At first, a 32x32 array was considered directly as a recognition space (input field), constructed from 64x64 original pixel array by a logical OR operation on groups of 4 pixels, i.e., $B_{ij} = 1$ iff $(A_{2i,2j} + A_{2i+1,2j} + A_{2i,2j+1} + A_{2i+1,2j+1} > 0)$. Since there are links from each input unit to each hidden unit, the size of the resulting network was still very large. It took almost four times as many computations to train and test the network than when a 16x16 Hough input field used. Further the training results were quite poor. When trained with 19 training samples for the first consonant, the network gave us 13 training successes out of 19 training samples, and 18 test successes out of 98 test samples.

Since the network corresponding to the first scheme took a large amount of time to be trained, we tried to reduce the simulation time by decreasing the size of input field. So 16x16 arrays were obtained from the 32x32 arrays by the simple compression method described above. It reduced the simulation time to almost one-fourth of the first case because the number of links between the input field and the hidden layer was decreased by a factor of exactly four. In that case two experiments were conducted; the network was trained with (1) 19 training samples and (2) 98 training samples, for the first consonant. Although the time to train the network was reduced, the training quality decreased: (1) training success rate was 47% and testing success rate was 15% (2)

training success rate was 22%. These results show that we lose significant information by using such a simple process to reduce the size of the input field.

We used two different types of input data (size 16x16). In both types, we reduce all values less than the eleventh highest element (thresholding value, 'Thr') of Hough array to 0. Type 1 input data are binary values; elements larger than 'Thr' are assigned the value 1. For the second type of input data, the remaining 11 non-zero values in the Hough array are normalized (divided by their max-value) to minimize loss of information and retain a measure of their respective strengths. We trained the network using both input data types mentioned above. The results are listed in Figure 7. We also varied the teaching input strategy and learning rate. The detailed results for each component type are presented in Figures 8-11 for Input Data type 2.

	FC module	LC module	VV module	HV module
TYPE 1	40.8 (at 1400 cycles)	70 (at 300 cycles)	68.3 (at 300 cycles)	94.9 (at 100 cycles)
TYPE 2	63.3 (at 700 cycles)	87.5 (at 300 cycles)	100 (at 300 cycles)	100 (at 100 cycles)

TYPE 1: Input Data Type 1, non-bias teaching input, constant learning rate

TYPE 2 : Input Data Type 2, bias teaching input, varying learning rate

Figure 7 : Comparison of the training success rates

Our input space is very sparse, with about 4% occupancy (at most 10 non-zero values out of 256 points). In this case, biased teaching seems to be necessary to modify the corresponding weight in case of no correct output.

Conclusions

This paper presents our preliminary investigation on recognizing Korean characters using neural networks. Implementation of our Korean character-recognition algorithms based on neural networks provided excellent recognition. Our results indicate that we can recognize about 81% of the training samples and 73% of the tested samples. We believe that our method can be fine-tuned to give better performance. A next step could be to use the presence of circles in Korean characters. We can perform a Hough transform to detect circles in the characters and this can be used (along with the Hough transform to detect lines) in the input space to improve performance.

References

1. Y. H. Huh and H. L. Beus, "On-line recognition of hand-printed Korean characters", Pattern Recognition, Vol. 15, No. 6, pp. 445-453,1982.
2. J. K. Lee and J. H. Lee, "The analysis-synthesis method of coordinate table for character pattern recognition", Proceedings of KIS conference, Apr., 1981.
3. K. H. Lee, K. B. Eom and R. L. Kashyap, "Recognition of Korean characters", Proc. 30th Midwest Symp. on Circuits and Systems, Syracuse, N.Y., pp. 161-164, Aug., 1987.
4. K. H. Seo, "Korean character recognition by hierarchical stroke matching based on an analysis-by-synthesis method", Ph.D dissertation, Syracuse University, N.Y., 1989.
5. P. V. C. Hough, "Method and mean for recognizing complex patterns", U.S. Patent 3069654, 1966.
6. D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes", Pattern Recognition, Vol. 13, pp. 111-122, 1988.
7. Thomas Risse, "Hough transform for line recognition: Complexity of evidence accumulation and cluster detection", CVGI 46, pp. 327-345, 1989.
8. D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation", Parallel Distributed Processing, Vol. 1, Ch.8, 1986.
9. Masafuni Hagiwara, "Accelerated back propagation using unlearning based on Hebb rule", IJCNN-90-WASH-D.C. Vol. 1, pp. 617-620, Jan., 1990.

	total no. of data	no. of training samples	no. of training successes	no. of successes (test)	no. of failures (test)
gierk	24	8	8	20	4
nien	11	7	0	0	11
digut	10	5	5	7	3
liel	8	5	0	0	8
miem	8	5	4	4	4
biup	16	6	6	13	3
siot	16	7	7	12	4
ieng	14	5	5	6	8
ziut	12	7	7	10	2
chiut	6	3	0	0	6
kiuk	9	5	1	2	7
tigut	6	4	0	0	6
piup	10	6	4	5	5
hieng	7	5	5	6	1
sgierk	6	5	4	4	2
sdigut	6	4	2	2	4
sbiup	3	3	0	0	3
ssiot	5	3	0	0	5
sziut	6	5	3	3	3
Total	183	98	61 62%	94 51.4%	89

Figure 8: Simulation results of F.C. module

	total no. of data	no. of training samples	no. of training successes	no. of successes (test)	no. of failures (test)
a	21	10	10	16	5
er	34	12	12	32	2
ie	28	13	13	26	2
va	4	3	3	3	1
ver	6	3	3	5	1
Total	93	41	41 100%	82 88.2%	11

Figure 9: Simulation results of V.V. module

	total no. of data	no. of training samples	no. of training successes	no. of successes (test)	no. of failures (test)
ee	11	5	5	9	2
o	31	14	14	28	3
oo	36	14	14	36	0
yo	10	3	3	3	7
you	6	3	3	6	0
Total	94	39	39 100%	82 87.2%	12

Figure 10: Simulation results of H.V. module

	total no. of data	no. of training samples	no. of training successes	no. of successes (test)	no. of failures (test)
gierk	25	6	6	21	4
nien	20	4	4	19	1
digut	2	2	2	2	0
liel	5	3	3	4	1
miem	3	2	2	3	0
biup	1	1	0	0	1
siot	6	4	4	6	0
ieng	12	4	4	11	1
ziut	3	2	2	3	0
kiuk	1	1	0	0	1
tigut	3	1	1	2	1
piup	1	1	0	0	1
hieng	1	1	1	1	0
sgierk	3	2	2	2	1
ssiot	1	1	0	0	1
nienhieng	1	1	1	1	0
lielgierk	1	1	1	1	0
lielmiem	1	1	1	1	0
lielhieng	1	1	1	1	0
biupsiot	1	1	0	0	1
Total	92	40	35 87.5%	78 84.8%	14

Figure 11: Simulation results of L.C. module