On Distributed Representation of Output Layer for Recognizing Japanese Kana Characters Using Neural Networks

Kanad Keeni, Hiroshi Shimodaira,
and Kenji Nakayama

Reprint

Proceedings of the

International Conference on Document Analysis and Recognition

Ulm, Germany
August 18-20, 1997

IEEE
COMPUTER
SOCIETY

Washington ◆ Los Alamitos ◆ Brussels ◆ Tokyo

PUBLICATIONS OFFICE, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1314 USA

On Distributed Representation of Output Layer for Recognizing Japanese Kana characters Using Neural Networks

Kanad Keeni†, Hiroshi Shimodaira† and Kenji Nakayama‡
† School of Information Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Ishikawa 923-12, Japan
‡ Dept. of Elec. & Comp. Eng., Kanazawa University
2-40-20 Kodatsuno, Kanazawa 920, Japan

Abstract

This paper presents an automatic coding scheme for representing the output layer of a neural network. Compared to local representation where the number of output unit is p, the number of output unit required for the proposed representation is close to $\log p$. The output of seven different printers were used for evaluating the performance of the system. The proposed automatic representation gave the average recognition rate of 98.7 % for 71 categories.

1 Introduction

Neural networks architectures have sparked of great interest in recent years because of their intriguing learning capabilities. Several learning algorithms have been developed for training the networks and out of them Back Propagation [1] is probably most widely used.

Previous work performed on recognizing zip-codes, Cun 1989[2]-[3], has shown that a fair amount of generalization on complex tasks can be obtained by designing a network architecture that contains a certain amount of a priori knowledge about the task. In their work they have shown through several experiments that the number of free-parameters in the network influences the network's generalization performance. In their method they have used weight sharing techniques which works in the same way as Neocognitron network model [4]. Raveendran and Oomatsu [5] have shown that cascaded neural networks can be a powerful tool in case of noisy input images. They have used a network to extract the features of the characters and the output of that network is fed to another network which works as a classifier.

On the other hand, there are several researches on Kanji recognition using Neural Networks [6] - [8]. However, due to the enormous number of Kanji category, in most of the cases after preprocessing the rough classification is performed, and depending on the result of the rough classification the input data is fed to different modules where the output layer is represented locally i.e, one unit for each category.

However, if the number of category to be recognized is very large and the output layer is represented in a conventional (each output unit representing a single category) way, then the number of connection from the hidden to output layer would become extremely large and it would cause the need for much more storage/memory.

In the present approach the output layer is represented distributedly and it is designed according to the feature of the input image. In [9], it has been shown that some heuristic knowledge can also be employed for designing a network.

Kana characters which are a subset of Japanese Kanji characters are used for the evaluation of the proposed method. In reality, by having a cursory look at any Japanese document, it can be seen that the percentage of Kana characters appearing in any text is almost 50 %. Generally, Kana characters are much more simpler than the Kanji characters. So, if the Kana characters are recognized separately with high accuracy then it will reduce the load of classifying simple characters along with complex characters and upgrade the overall performance of a Kanji recognition system.

This paper is divided into 5 sections. The next section describes the database along with the preprocessing steps. The third section describes the proposed representation of output layer. The fourth section presents experimental results. Finally, the last section is devoted to conclusion and further researches.

2 Database

Dataset	Training set	Testing set
A	1,2,3,4,5,6	7
B	1,2,3,4,5,7	6
C	1,2,3,4,7,6	5
D	1,2,3,5,6,7	4
E	1,2,4,5,6,7	3
F	1,3,4,5,6,7	2
G	2,3,4,5,6,7	1

Table 1: Training and testing data

The database is provided by the IPEC of Tohoku university and it consists of 7 sets of 128 by 128 binary images of 142 (2 × 71) printed Kana characters with two different fonts (Gothic, Minchou). Here, each dataset is generated from a different printer and the

0-8186-7898-4/97 $10.00 © 1997 IEEE

details of the database are summarized in Table. 1 (The numerals stand for the name of the printers; 1: CANON, 2: IBM 5587, 3: NEC, 4: OKI Microline, 5: RICOH Script, 6: SPARC, 7: FUJI Xerox).

2.1 Preprocessing

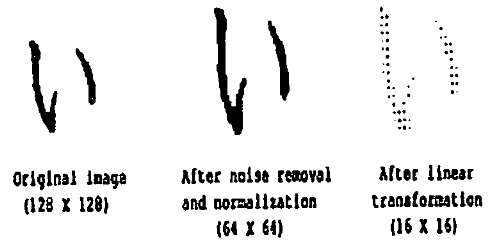

Figure 1: Preprocessing steps

The preprocessing is performed in three steps. First, extraneous marks are removed from the original image. Next, each character is normalized to a 64 by 64 binary image. Finally, a linear transformation is applied to reduce the image to a 16 by 16 pixel image. The reduction on the input image is performed by calculating the average of 4 × 4 pixels from the top left corner of the original image. Because of the linear transform, the resulting image is not binary but has floating point elements as the value of each pixel. The preprocessing process is summarized in Fig. 1.

3 Representation of Output Layer

In case of local representation p cells are required for representing p categories. However, if we can identify r critical features so that no two categories get the same features, then we could represent p objects as a pattern of r output cells and it would become a distributive representation. So, in contrast to local representation distributive representation would need less storage. For example, in case of representing 16 objects the local representations would need 16 output units. Whereas, in case of distributed representation, the output layer would be encoded by using only 4 ($\log_2 16$) units. This would need less storage and connections compared to the local representation. However, this kind of binary encoding is very sensitive to error. This is because if we invert the output of one unit, then we will get an ambiguous pattern that might be of 2 categories. This is also applicable to local presentation (because 2 values will be 1).

On the other hand, it is also well known that if we assign totally different code or teaching signal to similar objects then irrespective of the similarity the network would try to generate totally different weights for the objects and the learning would become complex.

In the present study, the above mentioned problems are considered carefully. In the present approach, the number of output unit lies closer to the $\log_2 p$ bound. Here, the basic idea is to code the output layer or the training signal in a way such that the similar characters get similar code or similar teaching signal.

3.1 Automatic coding scheme

The proposed automatic coding scheme is related to the input data and the method is as follows. At first each training data x_n $(n = 1, \cdots, N)$ (16 × 16 pixels) is split into four parts (8 × 8 pixels) in the following way.

$$x_n = (y_{n1}, y_{n2}, y_{n3}, y_{n4})$$

As a result the following set of pattern is generated for each part $i = 1, \cdots, 4$.

$$Y_i = \{y_{1i}, y_{2i}, \cdots, y_{Ni}\}$$

Next, each Y_i (i = 1, 2, 3, 4) is clustered. Here the LBG method [10] was applied for clustering and the number of cluster was set to 16. At this stage for each ω_l $(l = 1, \cdots, L)$ the cluster to which the part belongs is checked and the final cluster is determined by considering the maximum number of characters belonging to a specific cluster. So, in this way, for each category ω_l there will be a four dimensional vector $q_l = (q_{l1}, q_{l2}, q_{l3}, q_{l4})$ where each element of the vector will correspond to the cluster to which the corresponding part of the training sample belongs. Now, each element of vector q_l is mapped to a 5-bit binary pattern. The binary patterns are hand picked and they are mapped in a way such that for any two clusters whose parent is the same, the Euclidean distance between the binary patterns becomes one. Finally, q_l is transformed to a feature vector which is a 20-bit code. In this way for each training sample x_n $(n = 1, \cdots, N)$ there will be a feature vector $c_{(n)}$ with 20 elements. (Here (n) represents the category number to which x_n belongs.) The entire process is summarized in Fig. 2.

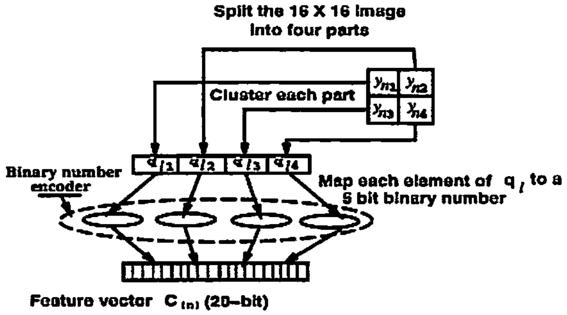
Split the 16 X 16 image
into four parts
Cluster each part
y_{n1} y_{n2} y_{n3} y_{n4}
q_{l1} q_{l2} q_{l3} q_{l4}
Binary number encoder
Map each element of q $_l$ to a 5 bit binary number
Feature vector C $_{(n)}$ (20-bit)

Figure 2: Automatic coding procedure

3.2 Recognition process

The recognition process is divided in to two parts. Rough classification and final classification.

VQ is performed for rough classification and the number of centroid was set to 3. After VQ, for each category ω_l $(l = 1, \cdots, L)$, the group to which the category belongs is determined by considering the first two candidates. In this way, the whole training set is divided in to three groups.

During testing, the module to which the character is to be fed is determined by calculating the Euclidean distance among the input vector x_n and the patterns belonging to each group G_i $(i = 1, 2, 3)$.

As described in the previous section, the automatic codes will provide a feature vector for each category. These feature vectors are utilized for representing the output units of the network.

The feature extraction procedure is performed by the feature extractor network which consists of one input layer, one output layer and a hidden layer. The network is totally connected. The output layer consists of P units.

After training, the network is fed with the training data and the output vectors for each category are stored. The *prototypes* are formed by calculating the average for each category and this process can be described by the following expression: $m_l = \frac{1}{N_l}\sum_{x\in\omega_l} o$. Here, N, o, and x stand for the number of samples used for each category during training, the output vector, and the input data respectively. These prototypes are later used for recognition.

During final classification, Euclidean distance is taken in between each prototype and the output vector given by the network during testing. The category for which the corresponding prototype gives the closest distance to the output vector is treated as the recognized category. The whole process, formation of orototypes to evaluation is summarized in Fig. 3.

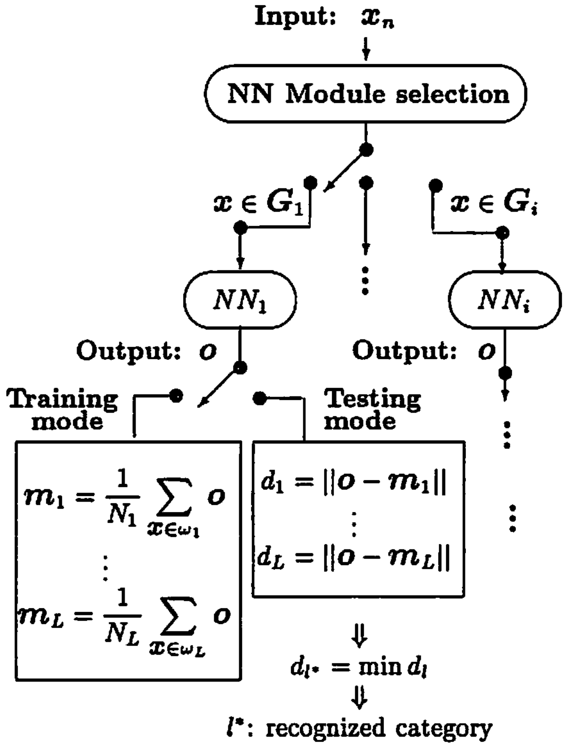

Figure 3: Training and recognition process

4 Experiments

After rough classification the remainder of the recognition is performed entirely by the proposed feature extractor network. The input to the network is a 16 × 16 floating point image. The output layer is composed of 20 units (20 dimensional feature vector). Here, all simulations were performed using the Aspirin/MIGRAINES Neural Network Software [10] running on SUN work station. The normal sigmoid transfer function used at each node calculates the output in the range of (0,1). Before training the weights are initialized with a random value with weights uniformly distributed in the range of (-0.1,0.1). The backpropagation learning algorithm is used in determining the ideal weights for all connections. The weights were updated according to the gradient method (averaging over the whole training set before updating the weights). Each training set contained 12 (6 × 2) samples per category and each test set contained 2 open samples per category. For each dataset, rough classification was performed as mentioned in Sect. 3.2 and in each case the centroid vetors were rebuild from the training data. The rough classification rate was 100 % for all datasets.

4.1 Results

Dataset	Accuracy (%)
A	97.9
B	100
C	97.2
D	100
E	100
F	95.8
G	100
Average	98.7

Table 2: Classification rate

Experiments were performed for each dataset by applying the proposed automatic coding scheme. The network had one input layer, one hidden and one output layer. A number of experiments were performed by setting the number of hidden units in between 10 and 45. And for each of them 5 different initial weights have been used. The network gave the best performance with 26 hidden units. The open Recognition rate for each dataset is summarized in Table. 2.

4.2 Relation of problem complexity and generalization

Dataset	Complexity	Accuracy (%)	No. of error
A	0.002494	97.9	3
B	0.000034	100	0
C	0.002110	97.2	4
D	0.0	100	0
E	0.0	100	0
F	0.010733	95.8	6
G	0.004512	100	0

Table 3: Problem complexity

In Table. 2, it can be seen that the network gave 100 % accuracy in respect to dataset B, D, E and G. And the accuracy rate for A, C and F were slightly dropped. However, it can be thought that the drop in the classification accuracy is related to the number data used during the open exeperiments and the complexity of the problem. Due to the small amount of testing data, even a single missclassification has largely effected the accuracy rate. In order to find the relation between problem complexity and gneneralization, the complexity of each problem $C_m (m = 1, 2, ...7)$ is calculated in the following way.

$$C_m = \frac{1}{K}\sum_k \frac{d(\boldsymbol{x}_k, \boldsymbol{T}_i) - \min_i d(\boldsymbol{x}_k, \boldsymbol{T}_i)}{\min_i d(\boldsymbol{x}_k, \boldsymbol{T}_i)}$$

Here, K, $d(\boldsymbol{x}_k, \boldsymbol{T}_i)$ stand for the number of testing data, and the Euclidean distance between the input vector $\boldsymbol{x}$ and template $\boldsymbol{T}_i$ $(i = 1, 2, ...N)$. The templates are formed by calculating the average of each category. In this way the complexity of each problem is calculated and they are summarized in Table. 3. From Table. 3, it can be said that the complexity of the problems are related to the generalization of the network. The generalization capacity of the network went down for datasets A, C and F. However, in case of B and G, irrespective of the complexity of the problems the network could classify the characters with 100 % accuracy. This assures that the proposed distributive representation generalizes well against complex problems.

In [11], experiments were performed for evaluating the effectiveness of the proposed automatic coding. In that case, any two clusters that had the same parent before they were split were given a pair of totally different codes i.e., one of them was kept the same as it was with the proposed coding scheme and the code for the other cluster was simply the inverted version of the former. It was found that the proposed coding results in better generalization.

The performance of the proposed automatic coding procedure could have been compared to the conventional output layer representation where each of the output unit represents a single category, however this is not considered for the present study. This is because, if the real world problems where the number of category to be recognized is too large (for example Kanji) are considered, then it is unrealistic to apply the conventional method which would need much more storage/memory than the proposed method.

5 Conclusion

We have classified 71 Kana characters with an average of 98.7 % accuracy. Although the small amount of testing data have largely effected the overall recognition rate, with more training and testing data the accuracy can be improved. In the proposed method it is not necessary to detect features as it was with [8]. In contrast to the conventional method, the proposed output unit representation reduces the number of connections and free parameters. By employing Kana characters which are a subset of Japanese Kanji character set, it has been shown that the coding scheme effects the generalization of the network. The next step would be to apply the present method for recognizing the whole Kanji character set.

The proposed automatic coding scheme was also applied in [11] for recognizing 41 Devanagari characters and it gave a recognition rate of 98.09 %. The present recognition accuracy of Kana character is also encouraging. Therefore, it is largely hoped that the same method would be extended for other classification problems.

Acknowledgments

The authors wish to thank the Tohoku university IPEC for providing the data. They are also grateful to Professor Masayuki Kimura of JAIST for his constructive criticism and thought-provoking remarks on the present work.

References

[1] Rumelhart, McClelland, and the PDP Research Group, "Parallel Distributed Processing," *The MIT Press*, 1989.

[2] Y. LeCun, "Generalization and Network Design Strategies," *CONNECTIONISM IN PERSPECTIVE, Elsevier Science Publishers B.V.(North-Holland)*, pp.143-155, 1989.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*,1(1989), pp. 541-551.

[4] K. Fukushima, "Neocognitron, A Hierarchical Neural Network Capable of Visual Pattern Recognition," *Neural Networks*, vol. 1, pp. 119- 130, 1988.

[5] Paramesran Raveendran, Sigeru Omatu, "Generated Moment Invariant Features by Cascaded Neural Network for Pattern Classification," *Transaction of Information Processing Society of Japan* , Vol. 35, No.2.

[6] Kenichi Hotta, et al., "A Large Scale Neural Network CombNET-II," Transaction of Information Processing Society of Japan, Vol. J 75-D, No. 3.

[7] Yoshimasa Kimura, et al., "Handprinted Kanji Pattern Recognition by a Neural Network which Unifies Multiple Methods by Processing Candidate Discriminants," *Transaction of Information Processing Society of Japan*, Vol.J77-D No. 4, 1994.

[8] Teruhiko Ohtomo. et al., "On Efficient Recognition of Hand-written Chinese Characters by Locally Connected Neural Networks," Transaction of Information and Systems Society of Japan, IEICE TRANS. INF. &SYST., VOL. E79-D, No. 5, May 1996.

[9] Kanad Keeni, Tetsuro Nishino, Hiroshi Shimodaira, "On Feature Extraction of Indian Characters by Using Neural Networks," *IEICE*, Technical report of IEICE, COMP94-24, pp. 1-9,1994-07.

[10] Russell R. Leighton, "The Aspirin/MIGRAINES User's Manual," Russel R. Leighton and the *MITRE Corporation*, Release V6.0, October 29, 1992.

[11] Kanad Keeni, Hiroshi Shimodaira, " Recognition of Devanagari Characters Using Neural Networks," Transaction of Information and Systems Society of Japan, IEICE TRANS.INF. &SYST., VOL. E79-D, No. 5, May 1996.

[12] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Comm.*, COM-28:84-95, January 1980.