DESIGN AND EVALUATION OF A HIERARCHICAL ARCHITECTURE FOR HANDWRITTEN CHARACTER RECOGNITION

Ioan Z. MIHU, *"Lucian Blaga" University of Sibiu, Computer Science Department, str. Emil Cioran, nr. 4, Sibiu, ROMANIA, email: ioan.z.mihu@ulbsibiu.ro*
Horia V. CĂPRIȚĂ, *"Lucian Blaga" University of Sibiu, Computer Science Department, str. Emil Cioran, nr. 4, Sibiu, ROMANIA, email: horia.caprita@ulbsibiu.ro*

Abstract:
Handwritten character recognition represents a problem that was approached in many ways by the scientists. Although a generally solution for any type of handwritten characters not founded yet, the obtained results give hopes to continue the researches in this field.

In this paper, we present a software architecture used in character recognition: Hierarchical Neural Network (HNN) architecture (Halici et al. 1999). Using this architecture, we have approached two important aspects from the handwritten character recognition task: input data preprocessing and clustering. The input data preprocessing consists of a compression applied to the input character in order to obtain a reduced standard character matrix. The reduced character matrix will be used, in an efficient manner, as input data, in the character recognition application. The first stage in the character recognition is the clustering stage performed by a Self-Organizing Feature Map (SOFM) Neural Network (NN). The clustering process consists in fact in a classification based on the input vectors similarities.

***Keywords**:* Hierarchical Architecture, Neural Networks and Handwritten Character Recognition

1. INTRODUCTION

The first step performed in a character recognition process is the common features extraction that could be viewed as a transformation of the input space into the features space.

The second step performed in a character recognition process is the classification. This process consists of two successive stages (Mihu et al. 2001):

- the preclassification
- the classification of the previous results

2. HNN ARCHITECTURE

The fourth hierarchical levels of HNN architecture presented in figure 1 are: the preprocessor level, the SOFM level, the recognition modules (PCA and MLP) and the voting mechanism. The preprocessor performs the transformation of the input data space into the features space.

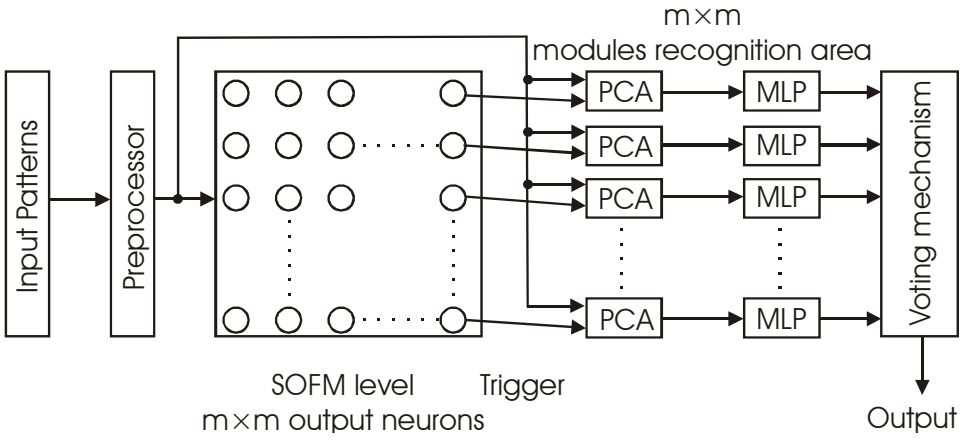


Figure 1. HNN architecture (Halici et al. 1999)

2.1. The Preprocessor

The preprocessing stage consists of two steps:

- character selection
- character compression

The preprocessor's inputs are bitmap files obtained by scanning a handwritten document page. We designed a graphical selector controlled by user by means of mouse device (figure 2).

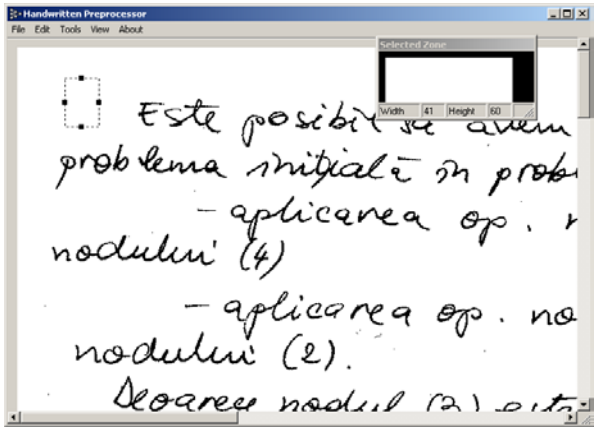


Figure 2. The graphical selector:

2.2. Character Normalization Algorithm

In the character selection stage it is obtained a m×n character matrix (m, n > 10) (Mihu et al. 2001). The SOFM module disclaims standard compressed vectors as inputs.

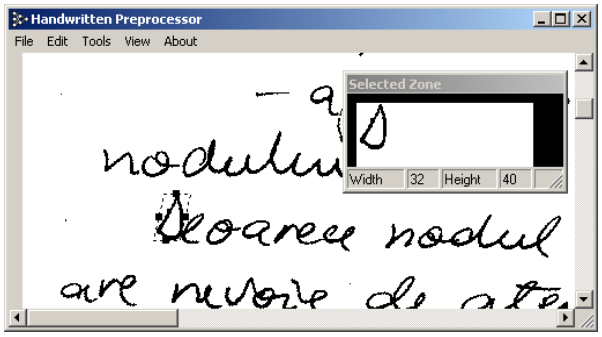
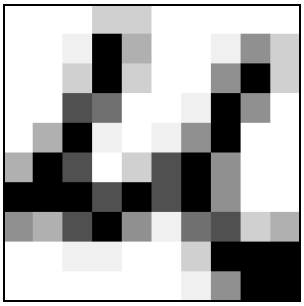


Figure 3. Character 'D' selection (40×32 pixels)

The normalization algorithm will compress the selected character and generates the standard 10×10 character matrix for the SOFM module (figure 4).

In these matrices, a 0 value represents the black color and a maximum value represents the white color; the intermediary values are levels of gray. Figure 8 shows a standard character generated by the normalization algorithm and his representation in gray levels ("8-grays" matrix). The "n-grays" matrices were used in the simulation process in order to take a decision about the best matrix format that could be used for the HNN architecture.



7775577777
7760477635
7750577305
7712776037
7406763077
4017510377
0001010377
3410362154
7766775000
7777776300

Figure 4. The graphical representation of 'u' character and the afferent matrix resulted after normalization (8-grays)

3. THE HNN SIMULATION

In the first step, it is necessary to perform a verification process. For that, we followed a simple procedure:

- The bitmap files (used as inputs for the preprocessor) were built from printing characters written with two different fonts: *Courier New* and *Times New Roman*;
- In the learning phase of the SOFM, the training vectors were characters from the *Courier New* font;
- In the recognition process, the input vectors were characters from the *Times New Roman* font;

In the second step of the verification process, we used handwritten characters by different people.

4. THE EVALUATION RESULTS

The results presented in this paragraph refer to the preprocessor and the SOFM levels:

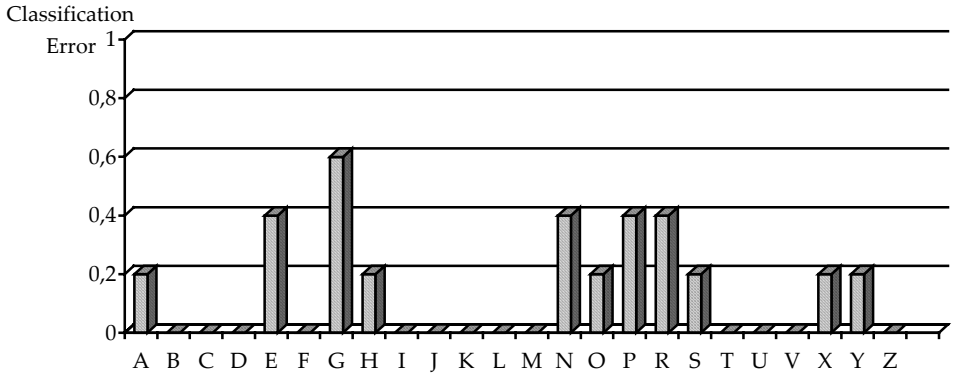
- 4×4 SOFM network, "8-grays" input vectors;
- 6×6 SOFM network, "8-grays" input vectors;

4.1. The results corresponding to 4×4 SOFM architecture

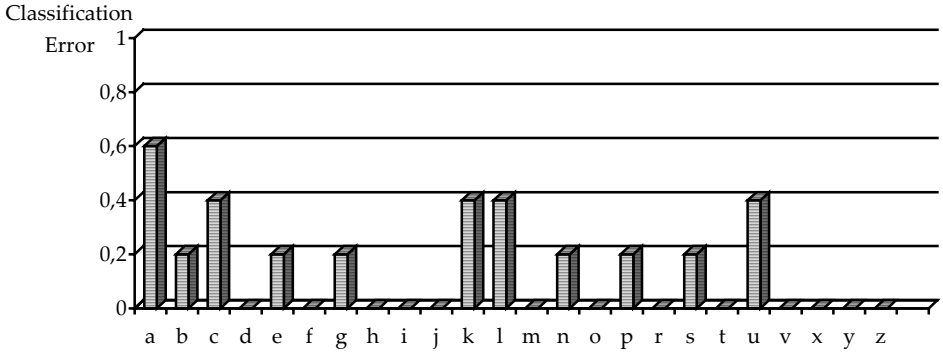
The clusters configuration is presented in table 1 and the classification error is presented in the histogram in figure 5.

Table 1. The obtained clusters using "8-grays" input vectors

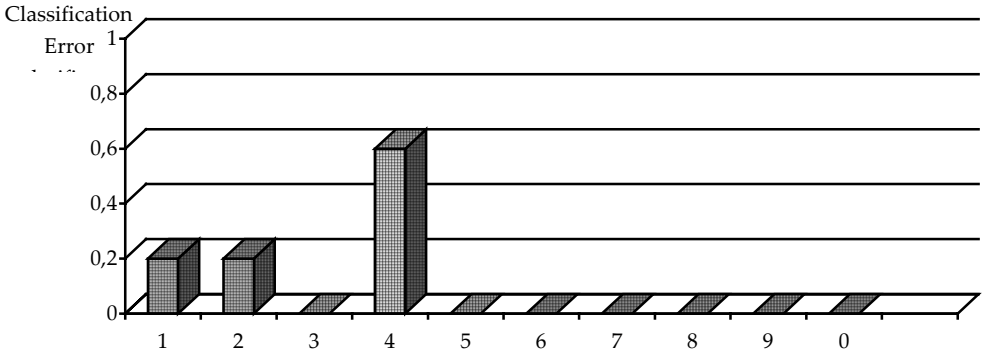
ID Cluster	Assigned Character
0	X, x
1	N, h,r,4
2	K, R, k
3	G, U, g, 9
4	V, Y
5	F, I, T, f, v,5
6	C, c, e
7	M
8	J, j, y, 7
9	L, Z, d, i, t, z,2
10	E, S, 6, 8
11	a, m, n, u
12	1
13	B, D, b, l, s, 3
14	O,o,0
15	A, H, P, p



a) the classification error for the upper characters



b) the classification error for the lower characters



c) the classification error for the numeric characters

Figure 5. The classification error for 4×4 SOFM network, "8-grays" input vectors

5. CONCLUSIONS

The advantages of the implemented character normalization algorithm are:

- The compression doesn't affect the essential features of the characters. The resulted matrix retains, in a compressed form, the character with his essential features (see figure 6);
- The compression level is automatically adjusted. Starting from source matrices of variable dimensions, the algorithm will generate destination matrices of standard dimension (10×10 pixels);
- The primitives libraries (see figure 6) obtained by using the compression algorithm can be used as training sets for the neural networks in the next levels of HNN in the character recognition application.

The obtained results led to the following conclusions:

- The HNN architecture is very efficient for printed character recognition task;
- The performances decrease in case of handwritten character recognition. The errors are inherent in this stage (because of input patterns diversity); they will be eliminated by the following modules in the HNN architecture (PCA, MLP and voting mechanism).

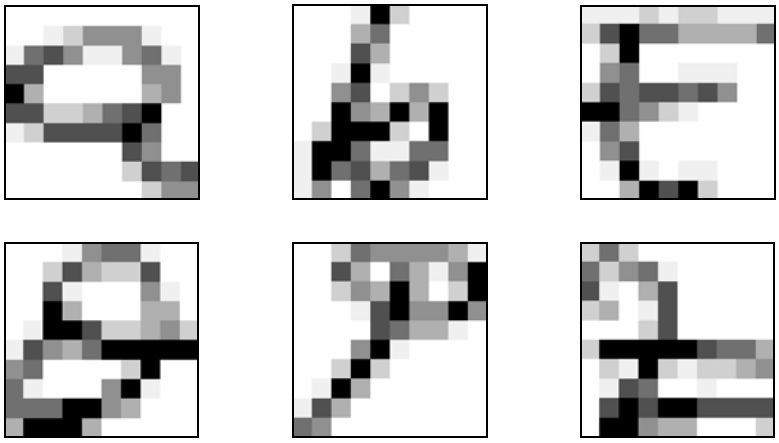


Figure 6. The a, b, E, o, p, z primitives obtained in preprocessing stage ("8-grays" input vectors)

REFERENCES

Bennani Y., Fogelman-Soulie F., P. Gallinari, "Text - dependent speaker identification using LVQ", Proc of the Int. Neural Networks Conf., vol. 2, Paris, July 1990, pag. 1087 - 1090.

Comon P., Voz J. L., Verleysen M., "Estimation of performance bounds in supervised classification", In M. Verleysen, editor, ESANN: European Symposium on Artificial Neural Networks, Bruxelles, April 1994, pag. 37 - 42.

Guyon I., "Neural networks and applications", Internal Report AT & T Bell Labs. September 1990.

Halici U., Erol A., Ongun G., "Industrial Applications of Hierarchical Neural Networks: Character Recognition and Fingerprint Classification", Industrial Applications of NNs, CRC Press, 1999, pag. 159-192.

Kohonen T., "The "neural" phonetic typewriter", Computer, March 1988, pag. 11 - 22.

Kohonen T., "Self - Organization and Associative Memory", Berlin, Springer - Verlag, 1988.

Kohonen T., "Self - Organizing Maps", Springer - Verlag, Berlin Heidelberg, 1997.

Mihu I. Z., Căpriță V.H., "Preprocessor for the Handwritten Character Recognition" (in romanian), Proceedings of ATU, Sibiu, 2001.

Schalkoff R.J., "Artificial Neural Networks", McGraw-Hill, 1997.

Zurada J.M., "Introduction to Artificial Neural Systems", West Publishing Company, St. Paul, 1992.