MULTIFONT OTTOMAN CHARACTER RECOGNITION

Ali ÖZTÜRK[1], Salih GÜNEŞ, Member, IEEE[2] and Yüksel ÖZBAY[2]

[1] Bilkent University, Computer Center, 65530, Ankara/TÜRKİYE
[2] Selçuk University, Electrical and Electronics Engineering , 42031, Konya / TÜRKİYE
e-mail: ozturka@bilkent.edu.tr, sgunes@selcuk.edu.tr, ybay@selcuk.edu.tr

ABSTRACT - Ottoman characters from three different fonts are used character recognition problem, broadly speaking, is transferring a page that contain symbols to the computer and matching these symbols with previously known or recognized symbols after extraction the features of these symbols via appropriate preprocessing methods. Because of silent features of the characters, implementing an Ottoman character recognition system is a difficult work. Different researchers have done lots of works for years to develop systems that would recognize Latin characters. Although almost one million people use Ottoman characters, great deal of whom has different native languages, the number of studies on this field is insufficient.

In this study 28 different machine-printed to train the Artificial Neural Network and a %95 classification accuracy for the characters in these fonts and a %70 classification accuracy for a different font has been found.

1. INTRODUCTON

We can describe the character recognition operation as transferring the hand-written or printed text to the computer correctly to facilitate the human-computer interaction. Segmentation of the symbols in document analysis systems is the most critical operation. The recognition of a hand-written Text is more challenging that recognizing a printed text. For this reason the recognition applications of printed characters are implemented much more. Often, the written or printed symbols that will be recognized may be interconnected. While the Latin characters are interconnected if they are hand-written, the Ottoman characters are being found as interconnected character groups even in a printed text. The noise which appears not only from the natural effects of hand writing, but also by means of other factors such as scanner resolution, printer and paper quality may connect the symbols or breaks one symbols into pieces. This problem is known as connectivity and segmentation.[1].

The neural network that was used to recognize the Ottoman Characters in this study is a MultiLayer FeedForward Network with BackPropagation. There are 700 nodes in the input layer for each character, 30 nodes in the hidden layer and 28 nodes in the output layer for each characters. Sigmoid function is used as activation function.

2. THE PROPERTIES OF THE OTTOMAN CHARACTERS

Although important achievements are made in recognizing the Ottoman numbers, there are very few papers about recognizing the Ottoman characters especially by means of neural networks. Recognizing the Ottoman characters is a challenging work because of the evident features of them. Each character has four different shapes according to its position in a word. In a word a character may be :

- At the beginning
- In the middle
- At the end and
- May be isolated from other characters.

The properties of an Ottoman text that makes it different from all other languages are [2] :

- The characters are as interconnected symbol groups. (To be hand-written or printed doesn't make sense).
- The shape of a character changes according

0-7803-6542-9/00/$10.00 © 2000 IEEE

to the position in an interconnected character group.

- The width of the characters may be different even for the same font and the width of a character may change according to its position.
- Some characters may be vertically aligned.

As either the hand-written or the printed Ottoman text includes interconnected character groups, the segmentation operation is very important. Even to recognize a printed Ottoman text is more valuable than recognizing a hand-written Latin text, because the character width may change in different positions[3].

Table 1. Basic Ottoman characters each of which belongs to a different class

Class No	Letter Group	Class No	Letter Group	Class No	Letter Group
1	ا	8	ظ ط	15	ه
2	ث ت ب	9	غ ع	16	و
3	ج چ ح	10	ق ف	17	ي
4	ذ د	11	ك		
5	ز ر	12	ل		
6	ش س	13	م		
7	ض ص	14	ن		

Table 2. The basic character groups that are obtained by assigning the characters having similar shapes to the same group

Class No	Letter	Class No	Letter	Class No	Letter	Class No	Letter
1	ا	8	د	15	ض	22	ك
2	ب	9	ذ	16	ط	23	ل
3	ت	10	ر	17	ظ	24	م
4	ث	11	ز	18	ع	25	ن
5	ج	12	س	19	غ	26	ه
6	ح	13	ش	20	ف	27	و
7	خ	14	ص	21	ق	28	ي

3. BACKPROPAGATION NEURAL NETWORK

Multilayer perception trained by backpropagation [4] is the most common and famous neural network classifier. It is well known that the networks having one hidden layer and non-linear activation function are the most widely used classifiers. If they are trained properly they have the power to make the right classification after obtaining the property set which determine a pattern as an input vector from their input layers.

After feeding the input vector to the input layer of the neural network, passing through the hidden layers by means of weight values, the result appears in the output layer. Each of the neuron in the network passes the value to all the neurons behind its layer as the result of a non-linear activation function after taking the arithmetical sum of all the weight values that end on itself. This explanation can be summarised as in the following equation.

$$out_i = f(net)_i = f(\sum_j w_{ij} * out_j + \theta_i) \qquad (1)$$

There are many alternatives for an activation function. In this study "Sigmoid" is used.

$$f(net_i) = \frac{1}{1 + e^{\frac{-net_i}{Q_0}}} \qquad (2)$$

Q_0 is the temperature value of the neuron. The higher the temperature, the slower the sigmoid function changes. In very low temperatures it becomes a step function [4].
In the backpropagation-training algorithm, the mean square error is used as the error criterion.

$$E_p = \frac{1}{2}\sum_{j=1}^{N}(t_{pj} - Opj)]^2 \qquad (3)$$

Here E_p is the error for the p^{th} vector, t_{pj} is the expected value for j^{th} neuron (that is the related output value in the training set) and O_{pj} shows the real value of the j^{th} output neuron. Taking the square of the errors provides the output values that are far from the expected values to constitute the total error. If we increase the exponent this effect will be much strong.

The equation used for backpropagation training is,

$$\Delta w_{ji} = \eta \delta_{pj} O_{pi} \quad (4)$$

Here η is the learning rate, δ_{pj} is the error signal of the neuron in the L^{th} layer ve O_{pi} is the output value of the neuron in the $(L-1)^{th}$ layer . δ_{pj} error signal,

1) For the output layer neurons,

$$\delta_{pj} = (t_{pj} - O_{pj}) O_{pj} (1 - O_{pj}) \quad (5)$$

2) For the hidden layer neurons

$$\delta_{pj} = O_{pj} (1 - O_{pj}) \sum_k \delta_{pk} w_{kj} \quad (6)$$

Here O_{pj} is for the L^{th} layer neurons, O_{pj} is for $(L-1)^{th}$ layer neurons and δ_{pk} is for $(L+1)^{th}$ layer neurons.

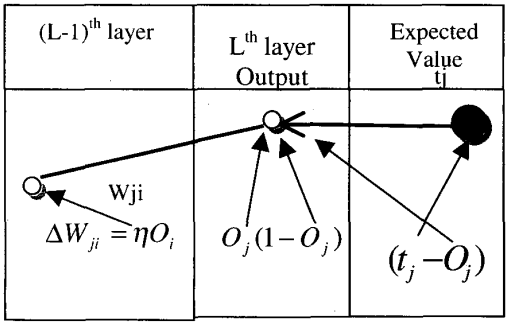

Figure 1. Output layer neurons training

In practice a coefficient α is added to the equation 4 in order to make faster convergence . This coefficient value provides to obtain the previous values of the weights and makes the error surface of the weigth space smooth filtering the changes in high frequencies. The weight values are adjusted as in the following if a momentum constant is used:

$$\Delta W_{ji}(n+1) = \eta(\delta_{pj} O_{pi}) + \alpha \Delta W_{ji}(n) \quad (7)$$

The momentum constant value is generally 0.7, but it can change for a specific application. The optimum value can only be found by means of experiments.

To begin training with the backpropagation algorithm very small values are given to the weights. Making opposite of this may cause the training to fail. In the next step the training set vectors are applied to the input layer neurons.

After the forward computation the real values will occur on the output layer neurons. The weight values are adjusted by applying backpropagation algorithm. The total network error has to decrease after much iteration. If this is not the case, the training parameters η ve α may be changed. Including contradictory values in the training set (e.g., giving different expected output values for the same input vector) may also cause the total error not to decrease independent from the training parameters. In this case the training set must be checked.

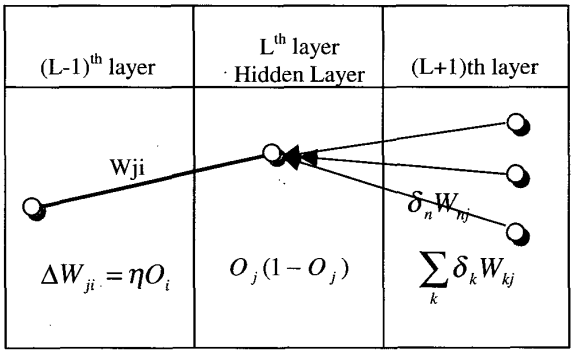

Figure 2. The training of the hidden layer neurons.

If the total network error falls below a predetermined value after a number of iterations then it means that the network has converged. Decreasing of the total network error linearly is not needed. There may be oscillation in the very first steps of the training. [1]

4. CHARACTER RECOGNITION SYSTEM

There are two main steps for the system implemented in this study:

- Training the neural network
- Testing the system

The data must be converted to the form that the neural network will be able to process and the

output values must be meaningful to the system user (the human). These operations are called preprocessing and postprocessing respectively.

4.1. Preprocessing Step

The data that is needed to train the neural network, which is the classifier part of the system, are obtained from three different Ottoman newspapers that make publications via the internet(Figure 3). Since these texts were written via keyboard and didn't contain any noise, the preprocessing step was relatively easy. These texts are published as images in GIF picture format on the newpapers homepages. These were converted to 4 bit BMP images by means of Microsoft Photo Editor , because 16-colour image is sufficient to handle the black-white text images[5][6]. The Ottoman characters in these images are manually segmented and pasted on a different BMP image file. To be processed by means of the neural network, these characters must passed through segmentation and feature extraction step.

4.2. Segmentation

To express the characters digitally, the BMP file must be converted to a text file that includes zero and ones. The program that is written for this purpose provides this situation by writing 1 for black pixels and 0 for white pixels.

Figure 3. The Ottoman characters in three fonts that are used to train the neural network

The segmentation operation finds the upper, lower, left and right bounds of each character in the text file and stores them as 20x35 matrixes. This matrix size is the optimum value that can be used without loosing the features of the characters. The following steps are applied to extract the characters from the file :

1) Since the characters are written line by line on a page the algorithm finds the line bounds on its first pass.
2) The left and right bounds of each letter are extracted from each line. (Figure 4)

3) The character data that are stored as zeros and ones in matrixes are normalized for the input layer of the neural network. That is, the two dimensional matrix of 20x35 elements is converted to a one-dimensional array of 700 elements.

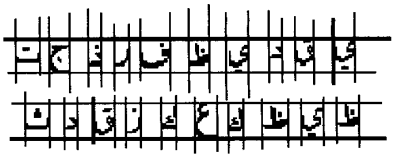

Figure 4. Segmentation of the Ottoman characters.

The characters data extracted from the steps explained above and the corresponding output values were written to a file that would be used for training. Since there are 28 basic characters 84 different character vectors are obtained from 3 different fonts. Since there are 700 data each of which is on a different line for each character, there are 58800 lines for the input data and a matrix of 28 column and 84 rows for the corresponding output values in the training file.

4.3. Training The Neural Network

BORLAND C++ for Windows writes the program that implements the supervised feedforward neural network with backpropagation. The program requires the following values from the user:

a) The name for the file that will keep the training results
b) The name of the file which includes the training data
c) The neuron number in the input layer
d) The neuron number in the output layer
e) Hidden Layer number and the neuron

number in the hidden layers
f) Maximum iteration number
g) Learning Rate (Eta)
h) Weight Adaptation Rate (Momentum)

The program has also the ability of interrupting the training, saving the weights and continuing from the point where it remains. At the end of training the program saves the data about the network state to the related files.

4.4. Testing The Recognition System

To test the implemented character recognition system and to give visual properties to the program a graphical user interface was developed using Delphi programming language. In this interface, there are two main windows on the interface. One of them is the window that contains the characters that will be used to test the system and the other is a small box in which the results are displayed. The user selects the characters via mouse and gives the command "Recognize" to the system. The pronunciation of the selected characters appears on the screen.

5. SIMULATION RESULTS AND CONCLUSIONS

The input layer of the neural network contains as many neurons as the row numbers (700) of the input vectors and the output layer includes a neuron for each of the 28 Ottoman characters. There are 30 neurons in the single hidden layer. The neural network was trained for 8 hours with nearly 4000 iterations until the total network error decreased to 0.000106. The network has achieved to classify the characters in the training set by nearly %94 classification accuracy. Only the following characters are misclassified.

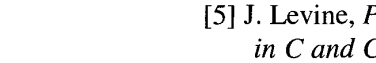

Figure 5. The misclassified training set characters

Since there are 28 basic characters in Ottoman alphabet, 84 different character vectors are obtained from 3 different fonts. The matrix that stores a character includes 20 rows and 35 columns, so the input layer must have 700 neurons. The output layer has 28 neurons for each character. To determine the misclassification rate for each character the following Root Mean Square Error equation was used.

$$E(f) = \left[\frac{1}{N} \sum_{k=1}^{N} \left| t\arg et - found \right|^2 \right]^{\frac{1}{2}} \qquad (8)$$

The character recognition system has been tested with a different font character set which doesn't contain any characters from the training set and found a %70 classification accuracy for the characters in this new font.

6. REFERENCES

[1] S. P. Abhijit, B. M. and Robert, *Pattern Recognition with Neural Networks in C++*, IEEE Press, 1996.

[2] B. Al-Badr, R.M. Haralick, " *Segmentation-Free Word Recognition with Application to Ottoman*", The Intelligent Systems Laboratory, University of Washington, 1995.

[3] R.F. Walker, M. Bennamoun and B. Boashash, "*Comparative Results For Ottoman Character Recognition Using Artificial Neural Networks*", Queen Sland University of Technology, 1995.

[4] D.E. Rumelhart, G.E. Hinton and R.J. Williams , "*Learning Internal Representations by Error Propagation*", Parallel Distributed Processing, vol.1, Foundations, D.E. Rumelhart and J.L. McClelland (Eds.), MIT Press, Cambridge, MA, 1986, pp 318-362.

[5] J. Levine, *Programming for Graphics Files in C and C++*, John Wiley & Sons, 1994, New York.

[6] S. Rimmer, *Supercharged Bitmapped Graphics*, Windcrest/McGraw, 1992, NewYork.