Arabic Text Recognition

Ramzi Haraty and Catherine Ghaddar
Lebanese American University, Lebanon

Abstract: *The issue of handwritten character recognition is still a big challenge to the scientific community. Several approaches to address this challenge have been attempted in the last years, mostly focusing on the English pre-printed or handwritten characters space. Thus, the need to attempt a research related to Arabic handwritten text recognition. Algorithms based on neural networks have proved to give better results than conventional methods when applied to problems where the decision rules of the classification problem are not clearly defined. Two neural networks were built to classify already segmented characters of handwritten Arabic text. The two neural networks correctly recognized 73% of the characters. However, one hurdle was encountered in the above scenario, which can be summarized as follows: there are a lot of handwritten characters that can be segmented and classified into two or more different classes depending on whether they are looked at separately, or in a word, or even in a sentence. In other words, character classification, especially handwritten Arabic characters, depends largely on contextual information, not only on topographic features extracted from these characters.*

Keywords: *Arabic text classification, artificial neural networks.*

Received May 18, 2003; accepted July 24, 2003

1. Introduction

Artificial Neural Networks or ANNs have been successfully applied to many areas of pattern recognition, especially in the field of character recognition. Some researchers have used conventional methods for segmentation and recognition, while others have used ANN-based methods for the character recognition process [1, 2, 4].

The main challenge with handwritten text recognition is the presence of lines, non-character objects, and noise or 'salt-and-pepper' in the scanned image. Characters can also be written in many different sizes, writing instruments (varying thickness and stroke quality), and slants (causing character shearing along the horizontal axis).

There are also several major problems with Arabic handwritten text processing: Arabic is written cursively and many external objects are used such as dots, 'Hamza', 'Madda', and diacritic objects. In addition, Arabic characters have more than one shape according to their position in a word. Classifications of Arabic texts largely depend on context basis since many characters can be classified into different classes depending on whether we look at them in a word or in a sentence. This makes the problem of classification even more challenging.

This paper describes a method to classify Arabic handwritten texts. The method comprises three main components. The first, a heuristic algorithm, which is responsible for scanning already extracted-segmented characters to extract features to be used in the second component. A conventional algorithm was used for the initial segmentation of the text into connected blocks of characters [8]. The algorithm then generates pre-segmentation points for these blocks. A neural network is subsequently used to verify the accuracy of these segmentation points [9]. Another conventional algorithm uses the verified segmentation points and segments the connected blocks of characters. The second component is a combination of two ANNs, which return a set of output classes to be fed into the third component. The third component is responsible for specifying the class representing input characters.

The remainder of this paper is as follows. Section 2 presents the difficulties and obstacles found when processing handwritten text. Section 3 describes our work. Section 4 presents the experimental results of this approach. Section 5 discusses related work, and finally a conclusion is drawn in section 6.

2. Character Classification Obstacles

There are several major problems with Arabic text recognition. They can be classified into three main categories: general difficulties, handwritten text specific difficulties, and Arabic text specific difficulties.

2.1. General Difficulties

The following difficulties are common among character recognition methods in general:

- Presence of lines and other non-character objects.
- Presence of noise or salt-and-pepper in the scanned image.

- Linguistic problems, i.e., if a dictionary is used as a spelling checker to improve the accuracy of the recognition process, then proper names, acronyms, or other words that are not likely to be in the lexicon will decrease the recognition accuracy.

2.2. Handwritten Text Specific Difficulties

The following difficulties are specific to handwritten text segmentation and recognition methods:

- Variety in character size; i.e., characters may be written in many different sizes without changing their meaning.
- Variety in writing instrument, i.e., characters may vary in line thickness, color, and/or stroke quality. A thick stroke causes the following problems:
  1. Touching characters.
  2. Holes in letters; e.g., ? or letters get filled up partially or completely
  3. Thin stroke or low contrast may result in broken characters.
  4. Gaps in the stroke may also cause a lot of errors.
- Different writers or even the same writer under different conditions may slant their letters differently; i.e., their handwriting undergoes a shear along the horizontal axis.
- Translation problems; i.e., characters are not always written in the same position relative to the enclosing borders of the scanned image.
- Presence of similar symbols, like (1) and (?.

2.3. Arabic Text Specific Difficulties

The following difficulties are specific to Arabic text segmentation and recognition:

- Arabic characters can have more than one shape according to their position in a word whether at the beginning, middle, final, or stand alone, as shown in Figure 1.
- Different writers or even the same writer under different conditions will write some Arabic characters in completely different ways, as shown in Figure 2.
- Other characters have very similar contours and are difficult to recognize especially when non-character and external objects are present in the scanned image. Figure 3 shows a list of such characters.
- Handwritten Arabic characters depend largely on contextual information. There are a lot of handwritten characters that can be classified into two or more different classes depending on whether you look at them separately, in a word, or even in a sentence. For example, 5 in Arabic looks exactly the same as (?) and (zero) in Arabic looks exactly the same as a dot '.'.

middle end beginning stand alone

Figure 1. The shapes that character ? takes according to its position in a word

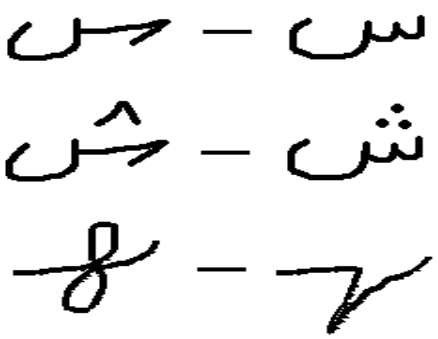


Figure 2. Three characters written in completely different ways.

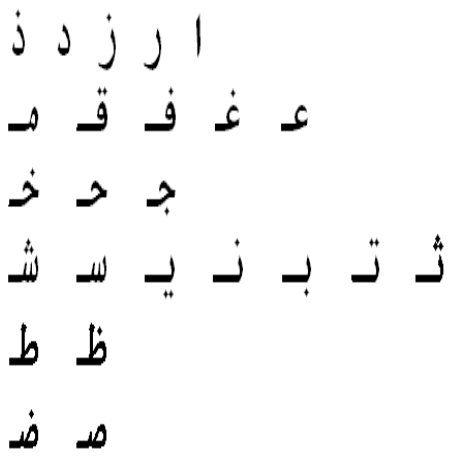


Figure 3. Characters with similar contours.

3. Neuro-Conventional Classification Method

There are a number of steps that need to be taken before a handwritten text can be recognized by a computer. These include sample data collection, analysis, scanning, binarization, segmentation, and classification. The main concern of this work is the classification part since [8, 9] investigated the binarization and segmentation issues.

3.1. The Data Set

Data sets of single characters in different shapes as they appear in a word were collected from students around the Lebanese American University as shown in Figure 4. These resulted in a 10027 different character samples used for training, and a 2132 different character samples used for testing. After preparing the scanned files, a manual classification process was followed to establish the desired output class of each character.

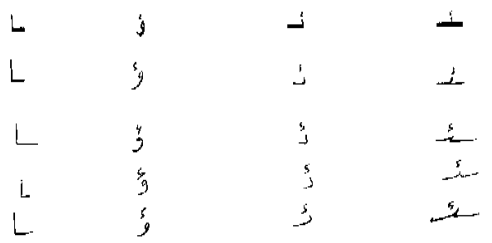


Figure 4. A sample of handwritten characters.

3.2. Binarization, Skeletonization, and Character Block Extraction

In [8], the authors describe a hybrid method to segment Arabic handwritten texts made of two components. The first is a heuristic algorithm, which is responsible for scanning a handwritten text, extracting blocks of connected characters, and then extracting features to be used in the second component. It is also responsible for generating pre-segmentation points, which are validated by the second component, the ANN. The ANN verifies whether all the segmentation points found are correct or incorrect.

Binarization, skelotonization, and character block extraction were adopted using [8]'s application, which converted the images into binary representations. A matrix of ones (1's) for black pixels and zeros (0's) for white pixels, then block character (BC) extraction, which is the extraction of more than one character written connected to another, forming a block of characters, was applied with a 94% accuracy. After that skeletonization, or thinning an image processing step that reduces BCs to their skeletons (i.e., transforming characters into arc segments one pixel thick, preserving connected components and the number of cavities and holes), was applied. The skeletonization process is required in order to extract certain features like corner points, end points, and fork points.

3.3. Feature Extraction

Feature extraction is the process of getting useful information from binarized files to be used for classification purposes. We were careful to extract features which are invariant and which capture the characteristics of a handwritten text by filtering out all the attributes that make the same character assume different appearances. Most, if not all, features used for the classification process were of the topographic type (i.e., point and area features). Each feature was represented by a number of attributes. These attributes quantify the nature of the feature. For example, specifying its position or size. An attribute may be represented as a continuous value, a discrete value, or a binary value.

Some systems attempt to learn feature extraction methods, starting only with a raw image, while others rely on sophisticated hand-coded feature extraction methods. Two important examples of learning feature extraction methods applied to handwritten character segmentation and recognition are the Karhunen-Loeve transformation [6], and the topographic feature maps obtained through weight sharing in the system described in [5, 8].

However, for Multi-Level Perceptron (MLP)-based systems [3], where each layer of the neural network feeds forward to all subsequent layers, hand-coded heuristic methods for extracting features such as endpoints, holes and corner points are a common and proven choice.

The main area features extracted were number of holes, number of corner points, and number of fork points from a binarized skeletonized image. The algorithm used for the extraction of these features is an adaptation of [8]'s work and changed as needed. A hole is an island of white pixels surrounded by black ones. An example of a hole is shown in Figure 5.

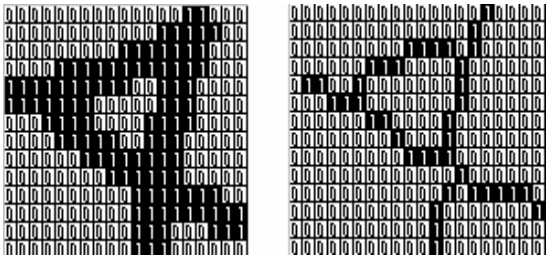


Figure 5. An example of a hole in a normal image (right) and a skeletonized image (left).

A corner point, on the other hand, is a black or white skeleton point that has at least two connected branches at right angles to each other, as shown in Figure 6.

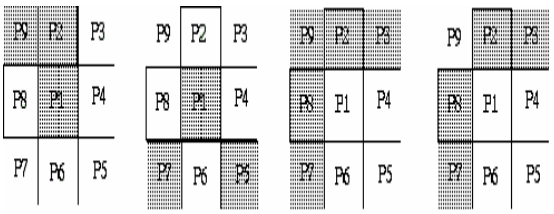


Figure 6. Four examples of corner points.

A fork point is a black skeleton point that has at least three connected branches as shown in Figure 7. In order to classify an image into its corresponding character, the features listed in Table 1 were extracted for each image.

Table 1. Classification features.

Feature	Attributes	Description
Width		Image width
Height		Image height
Image size		Width * height
Width-height ratio		Image width divided by height
Black pixels	Number of black points	Number of one's in image
	Number of black points divided by the height	Number of black points divided by the height
	Number of black points divided by the width	Number of black points divided by the width
	Black pixel density divided by (height * width)	Number of black pixels in the image divided by the total pixels in image
	Average column density divided by the height	Average column black pixel density divided by the image height
	Average row density divided by the width	Average row black pixel density divided by the width
Holes	Number of holes	Count the number of holes (or islands of white pixels completely surrounded by black pixels) in image
	Holes density	Number of black points of all holes divided by (width * height)
	Hole densities divided by the image density	Total number of hole pixels in image divided by the black point density
	Position of upper most, right most hole divided by the height	Y-coordinate of upper right most hole divided by divided by the height
End points	Number of endpoints in image	Number of endpoints in image
Corner points	Number of corners in image	Number of corner points in image
Fork points	Number of fork points in image	Number of fork points in image
Upper and lower contours	Index of highest pixel in upper contour divided by the height	Index of highest black pixel in upper contour divided by the height
	Index of lowest pixel in lower contour divided by the height	Index of lower pixel in black lower contour divided by the height
	Index of highest pixel in upper contour divided by the image density	Index of highest black pixel in upper contour divided by black point image density
	Index of lowest pixel in lower contour divided by the image density	Index of lowest black pixel in lower contour divided by the black point image density
Disconnected BCs	Number of disconnected BCs	Number of disconnected BC objects in image.
	Density of 1st disconnected BC	Number of black pixels in the BC divided by width*height.
	Density of 2nd disconnected BC	Number of black pixels in the BC divided by (width * height)
	Number of black pixels in BC1 divided by the image density	Number of black pixels in BC1 divided by the black point image density
	Number of black pixels in BC2 divided by the image density	Number of black pixels in BC2 divided by the black point image density

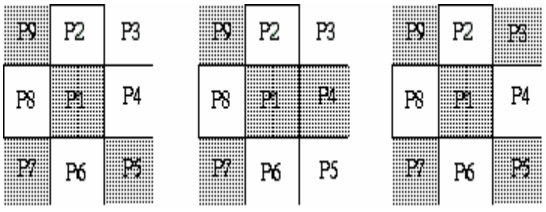


Figure 7. Three examples where P1 is a fork point.

3.4. Classification ANNs

The objective of the classification ANNs is to give an accurate output from the input features. To train the ANNs with both accurate and erroneous characters, the output from the binarization, skeletonization, and manual segmentation was used. It was necessary to manually specify the desired outputs for each character and save them to a file together with the extracted set of features.

In mathematical terms, the classification ANN is a set of functions which map inputs x_i to outputs y_i; where the outputs specify which of the classes the input pattern belongs to. There are 52 classes presented in Table 2.

Table 2. Classes of arabic images.

Classes	No. of Elements
Characters	28
Arabic Digits	10
Latin Digits	10
Ligatures (?)	1
Separators (“,”;”.”;”-“)	3
Total	52

3.4.1. ANN Architecture

Two generalized feed forward neural networks were used to give the desired output for characters. Each neural network is a generalization of the MLP such that each layer feeds forward to all subsequent layers. In theory, a MLP can solve any problem that a generalized feed forward network can solve. In practice, however, generalized feed forward networks often solve the problem more efficiently [5, 6].

There is no rule of thumb to determine good network architecture just from the number of inputs and outputs. It depends critically on the number of training cases, the amount of noise, and the complexity of the function or the classification of the network it is supposed to learn. There are cases with one input and

one output that require thousands of hidden units, and cases with a thousand inputs and a thousand outputs that require only one hidden unit, or none at all. To solve these issues many different networks with different number of hidden units were tried at first, starting with the smallest possible number of physical elements (PEs). The generalization error for each one was estimated, and the network with the minimum estimated generalization error that learned best to identify correct segmentation points was chosen.

An important criterion in ANN design is the network size. The number of PEs in a hidden layer is associated with the mapping ability of the network. The larger the number, the more powerful the network. However, if one continues to increase the network size, there is a point where the generalization gets worse. This is due to the fact that we may be over-fitting the training set, so when the network works with patterns that it has never seen before the response is unpredictable.

The best ANNs architecture reached consists of 26 inputs, 52 outputs, and 4 hidden layers. The 26 inputs were feature attributes of a character and the outputs were the 52 character classes. The architecture of the ANNs is summarized in Table 3.

Table 3. Architecture of classification ANNs.

	PEs	Transfer Function
Net #1		
Input Layer	26	Linear
Layer 1	26	Tanh
Layer 2	31	Tanh
Layer 3	37	Tanh
Layer 4	44	Tanh
Output Layer	52	Tanh
Net #2		
Input Layer	26	Linear
Layer 1	26	Tanh
Layer 2	30	Tanh
Layer 3	35	Tanh
Layer 4	41	Tanh
Output Layer	52	Tanh

The design of the classification ANNs described was implemented using NeuroSolutions, version 4.022 by NeuroDimensions, Inc. Each of the axons represents a layer, or vector, of PEs. All axons are equipped with a summing junction at their input and a splitting node at their output. This allows axons to accumulate input from, and provide output to, an arbitrary number of components. The Tanh axon used in these layers also applies a bias to each neuron in the layer. This will squash the range of each neuron in the layer to between -1 and 1.

The skeletonized and binarized character images were used to produce input files for the training, cross-validation, and testing phases of the ANN. 12159 characters were evaluated manually and divided into three parts as shown in Table 4.

Table 4. Manually evaluated input sets points.

Input Set	Number of Exemplars
Net #1	
Training set	10027
Cross-validation set	5014
Testing set	2132
Net #2	
Training set	10027
Cross-validation set	10027
Testing set	2132

Each character is represented by a row in the input files. Each row starts with the identifier of the image and its desired values, then the feature attributes are listed.

3.4.2 ANN Training and Testing

Training is the process by which the free parameters of the network, that is the weights, get optimal values. It is in fact a search in the so-called *weight-space*,or performance surface. This is the space spanned by all weights in the network. The goal of the search is finding a point in this weight-space which minimizes a certain error criterion [9].

The method used to train the classification ANN is the *back-propagation* method, a three step process:

Step 1: The input data is propagated forward through the network to compute the system output.

Step 2: The error between the desired and actual output is computed.

Step 3: This error is then propagated backward through the network, modifying weights on each layer until the first layer is reached.

Back-propagation modifies each weight of the network based on its localized portion of the input signal and its localized portion of the error. The change has to be proportional (a scaled version) of the product of these two quantities. The mathematics may be complicated, but the idea is very simple. When this algorithm is used for weight change, the state of the system is performing gradient descent; moving in the direction opposite to the largest local slope on the performance surface. In other words, the weights are being updated in the downward direction.

The advantage of using back-propagation is it is simple and easy to implement. The disadvantages are just as important: the search for the optimal weight values can get caught in local minima, i.e. the algorithm thinks it has arrived at the best possible set of weights even though there are other solutions that are better. Back-propagation is also slow to converge. In making the process simple, the search direction is noisy and sometimes the weights do not move in the direction of the minimum. NeuroSolutions solves a lot of these problems. It implements back-propagation of

the error in a secondary "plane" that sits on top of the axons and synapses. This is called the back-propagation plane.

Supervised learning requires a metric of how the network is doing. This metric is determined by calculating the sensitivity that a cost function has with respect to the network's output. This cost function, *J*, is normally positive, but should decrease towards zero as the network approaches the desired response. The literature has presented several cost functions, but the quadratic cost function, shown in equation (1), is by far the most widely applied.

$$J(t) = \frac{1}{2}\sum_i f(d_i(t) - y_i(t))^2 \qquad (1)$$

The L2Criterion component is a square error criterion and implements the quadratic cost function. The error reported is simply the squared Euclidean distance between the network's output and the desired response as shown in equation (2), where *d(t)* and *y(t)* are the desired response and network's output, respectively.

$$e_i(t) = -(d_i(t) - y_i(t)) \qquad (2)$$

The L2Criterion passes the computed error to the back criteria control. This control is designed to stack on top of the L2Criterion, and communicate the received error values from the I2Criterion with the back-propagation components to perform back-propagation.

The delta threshold transmitter controls the communication of the back criteria component based on the amount of error change between iterations. The back criteria is allowed to transmit the error value when the change between successive iterations crosses a specified threshold, chosen to be 0.0001. This threshold value can also be specified to change (i.e., incremented, decremented, or scaled by a constant) each time it is crossed.

4. Experimental Results

4.1. Classification Results

The classification process depends for its feature extraction on the heuristic algorithm implemented by [7] for extracting BCs, which achieved 94% accuracy. So, we can say there has been a small percentage of error that affected the ANN training file, which resulted from some BCs that had external child objects located at a far distance from the parent BC in which the heuristic algorithm assigned them to BCs that they did not belong to. Moreover, there is the case where external objects were located at approximately the same distance to more than one parent BC; thus, they were duplicated and a copy was assigned to each of those parent BCs. This resulted in an erroneous manual classification; that is specifying a different desired output, and as a result, the ANN classifier resulted in a different class than the initial one.

Figure 8 shows examples of miss-located points. These problems generally occur when people quickly write Arabic words. In the first word, the two points should be under the fourth BC only. However, the two points clearly cover the third, fourth and fifth BCs. In the second word, the point should be under the third BC, but it clearly appears to be under the fourth. Similarly, the two points should be located under the fourth BC, but they appear under the fifth. In the third word, a similar problem occurs: the point and two points belong to the first and fifth BCs; however, they appear under the second and last BCs.

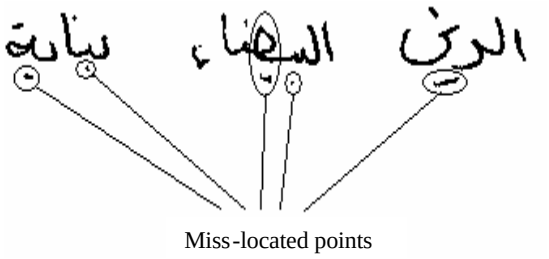


Figure 8. Three words containing miss-located external objects.

The output range of the ANN was between (–0.9) and (+0.9). A positive value indicated that the character may belong to the position class. A heuristic algorithm took the 52 outputs resulting from the ANNs, and compared these–thus selecting the highest value and specifying the position class related to it. There were some characters that were difficult to differentiate between their output results since these are of the same shape, and can not be differentiated except when they are seen contextually, that is in a word or a sentence. Table 5 shows these similar characters.

Table 5. Some similar objects.

Some Similar Objects
& ?
? & ?& 0
? & ?

4.1.1. Experimental Results of the ANN Classifier

Table 6 shows the results of the two classification ANNs trained on the 10027 training exemplars and tested on 2132 exemplars. Many experiments were performed varying settings such as the network type, the number of hidden layers and the number of processing elements in each layer. For each experiment, the number of inputs remained the same: 26 input features for each image.

Table 6. Classification ANN results using 10027 training exemplars, tested on 2132 exemplars.

ANN Architecture			MSE	Correct Characters
Network Type	Hidden Layers			
	No	PEs		
Feedforward MLP	4	26-31-37-44	0.62	791 (37.1%)
Feedforward MLP	4	26-30-35-41	0.65	778 (36.4%)

The ANNs in table 6 performed best in identifying correct characters. The minimum MSE achieved was 0.62. These two ANNs were used together–the feature file was passed to the first and then the second choosing the highest output value and thus returning the class corresponding to this index value. The ANNs were able to identify the accuracy of 1569 characters out of the 2132 character testing set; thus giving a recognition rate of over 73 percent. These networks had five identified characters in common. The first identified 15 different classes and the second identified 11 different classes–thus both identifying 26 different classes.

5. Related Work

A number of systems were developed to recognize Arabic text. One of these is TextPert 3.7 Arabic, produced by CTA Inc., which runs on the Macintosh Arabic system. Another is Al-Qari'al-Ali, a version of the program known as MULTREC, produced by Alamiah Software Co. Both of these programs were able to recognize certain computer printed texts of good quality with a reasonable degree of accuracy considering the difficulties of Arabic text [10].

Another system designed by Fehri and Ben Ahmed used a hybrid of Radial Basis Function Networks and Hidden Markov Models to recognize a printed Arabic text after identifying the used font. The results showed an increase in the recognition rate when the font is known prior to the segmentation process [11].

Sakhr developed Sakhr OCR for Arabic character recognition. The system uses an artificial neural network with a segmentation accuracy of 98% and a recognition accuracy of 99.8% for printed text [12].

6. Conclusion and Future Work

In this work, a classification process of handwritten Arabic text was presented as a division of three components, a heuristic algorithm to extract image features, two generalized feedforward networks to find the best output, and a classifier algorithm to specify the character corresponding to the index value returned by the network. They were used to classify difficult handwritten Arabic text, producing promising results. The two neural networks correctly recognized 73% of the characters. More testing and modifications will be conducted in the future hopefully resulting in the implementation of this technique to be used as part of a larger system.

Classification proved to be successful for some characters more than others especially those with a large number of training samples. A major challenge was encountered which is the similar contours of many Arabic characters and especially that we could not differentiate because the classification of an Arabic text depends largely on context basis; that is many characters can be classified into different classes depending on whether we look at them in a word or in a sentence.

As for future work, we plan to produce recognition that is context sensitive and to use a lexicon to improve proper output and somehow eliminate misclassification, thus integrating the different parts of the segmentation and classification into a complete Arabic handwritten recognition system.

Acknowledgements

We thank the anonymous referees whose comments helped in improving the presentation of the paper. We also thank Ms. Nabelah Haraty for her support.

References

[1] Al-Badr A. and Mahmoud A., "Survey and Bibliography of Arabic Optical Text Recognition," *Signal Processing*, vol. 41, pp. 49-77, 1995.

[2] Al-Yousefi H. and Upda A., "Recognition of Arabic Characters," IEEE *Pami*, vol. 14, no. 8, pp. 853-857, 1992.

[3] Almeida L., *Multilayer Perceptrons - Handbook of Neural Computation*, IOP Publishing Ltd and Oxford University Press, vol. 30, 1997.

[4] Bell J. and Zemanec P., "Test of Two Arabic OCR Programs," Distributed on Reader 14 January and Itisalat 17 January, 1995.

[5] Burges C., Ben J., Denker J., Lecun Y., and Nohl C., "Off Line Recognition of Handwritten Postal Words Using Neural Networks," *International Journal of Pattern Recognition and Artificial Intelligence*, pp. 689-704, 1993.

[6] Grother P. and Loeve K., "Feature Extraction for Neural Handwritten Character Recognition," *in Proceedings of SPIE*, pp. 155-166, 1992.

[7] Han K. and Sethi I., "Off-Line Cursive Handwriting Segmentation," *in Proceedings of ICDAR'95*, Montreal, Canada, pp. 894-897, 1995.

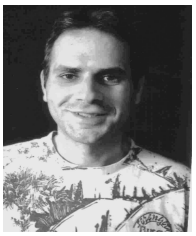
[8] Haraty R. and Hamid A., "Segmenting Handwritten Arabic Text," *ACIS International Journal of Computer and Information Science (IJCIS)*, vol. 3, no. 4, December 2002.

[9] Haraty R. and Zabadani H., "ABJAD: An Off-Line Arabic Handwritten Recognition System," *in Proceedings of the 2002 International Arab Conference on Information Technology (ACIT'2002)*, Doha, Qatar, December 2002.

[10] Khorsheed M. and Clocksin W., "Structural Features of Cursive Arabic Script," (*BMVC'99*), pp. 422-431, 1999.

[11] LeCun Y., Boser B., Denke J., and Jackel L., "Back-Propagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, pp. 541-551, 1989.

[12] Sakhr Software, www.sakhrsoft.com, 2003.



Ramzi Haraty is an assistant professor of computer science at the Lebanese American University in Beirut, Lebanon. He received his BSc and MSc degrees in computer science from Minnesota State University-Mankato, Minnesota, and his PhD in computer science from North Dakota State University-Fargo, North Dakota. His research interests include database management systems, artificial intelligence, and multilevel secure systems engineering. He has well over 50 journal and conference paper publications. He is a member of Association of Computing Machinery, Arab Computer Society and International Society for Computers and their Applications.

Catherine Ghaddar received her MSc degree in computer science from the Lebanese American University in Beirut, Lebanon. Her research interests include database management systems, neural networks, and software engineering.