Isolated Handwritten Arabic Characters Recognition using Multilayer Perceptrons and K Nearest Neighbor Classifiers

Yasmine Elglaly, Francis Quek
Computer Science Department, Virginia Polytechnic Institute and State University, USA

***Abstract*--In this paper we investigate the use of both the back propagation neural network and the k nearest neighbor (KNN) classifiers to recognize isolated handwritten Arabic characters. First we preprocess the input characters images, and then extract the main features, and finally we compare both classifiers in recognizing the training and the testing datasets.**

1. INTRODUCTION

Arabic characters are used in several languages, like Arabic, Persian, Urdu, Jawi and Pishtu, more than a half of a billion people use the Arabic characters [1]. Current handwritten Arabic character recognition systems do not give near hundred percent accurate results. More research in this area is still needed [2]. What makes Arabic characters more difficult to recognize than other languages is the cursive nature of Arabic writing which does not allow direct application of many algorithms designed for other languages. Furthermore, some Arabic characters are similar to each other [3].

We build a system that can classify off-line handwritten Arabic characters. The input to this system is a bitmap image containing only isolated handwritten Arabic characters, and the system should respond with the correct classification; where each class represents one character. First, the system begins by preprocessing the input image to normalize it. Then a feature extraction function is applied to the processed image. Finally we apply two classification algorithms that use the back propagation neural network and k nearest neighbor as classifiers.

In the next section, we survey the different existing Arabic characters recognition systems. In section 3 we describe the preprocessing steps and the algorithms we applied, an explanation of the extraction of features then we demonstrate the design of the back propagation neural network and kNN that are used in classification. Section 5 contains the experimental results and discussion. Finally, we state our conclusions in section 6.

2. RELATED WORK

There are two types of character recognition systems: on-line and off-line systems. Each system has its own algorithms and methods. The main difference between them is that in an on-line system the recognition is performed in the time of writing while the off-line recognition is performed after the writing is completed. AI-Muallim and Yamaguchi [4] proposed a structural recognition technique for Arabic handwritten words which were segmented into strokes. The strokes were classified and combined into characters according to their features. Amin and Alsadoun [5] proposed techniques using a binary tree to segment printed Arabic text into characters. Amin and Alsadoun [6] proposed recognition of hand printed Arabic characters using neural network. Aburaiba [7] dealt with some problems in the processing of binary images of handwritten text documents, such as extracting lines from pages, which is found to be powerful and suitable for variable handwriting. Abuhaiba et al. [8] introduced a novel offline cursive Arabic script recognition system to recognize offline handwritten cursive script having high variability based on segmentation based system. Khorsheed [9] presented a new method on off-line recognition of handwritten Arabic script, that is based on Markov model (HMM). HMM is also used in Alma'adeed et al [10] for unconstrained Arabic handwritten word recognition. In Alma'adeed [11], a complete scheme for unconstrained Arabic handwritten word recognition based on a neural network is proposed.

3. METHODOLOGY

This research that aims to build isolated handwritten Arabic characters recognition system is conducted through three main modules. The first module is responsible for preparing the input images by removing noise, converting images to black and white, and normalizing the size of the images. The second module extracts the main features of the preprocessed images. The third module processes the main features to recognize the input characters. Two methods are used in this module; the back propagation neural network and the k nearest neighbor algorithm.

A. *Preprocessing*

1. *Remove noise*

The first thing we perform on the input image is to remove noise and smooth the whole image. This is because the presence of undesirable dark pixels in the image gives rise to errors during character recognition.

We choose to implement the low pass Gaussian filter [12] with a small sigma value. This step successfully eliminates the noise in the input images, and keeps the structure of the character especially the complementary parts of the character that we need to differentiate similar characters. Figure 1, it shows an image for a character that is distorted with noise, and the image after being filtered.

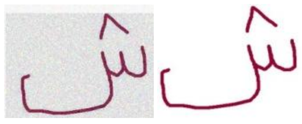
Figure 1. (a) image with noise (b)filtered image

The Gaussian filter is created using the following equations:

$$h_g(n_1,n_2) = e^{-(n_1^2+n_2^2)/(2\sigma^2)}$$

$$h(n_1,n_2) = \frac{h_g(n_1,n_2)}{\sum_{n_1}\sum_{n_2} h_g} \qquad (1)$$

2. *Binarizing*

The aim of this step is to convert any input color or grey images into black and white images. We use Otsu's method [13] to automatically perform histogram shape-based image thresholding. The algorithm assumes that the image to be thresholded contains two classes of pixels (e.g. foreground and background) then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal.

We exhaustively search for the threshold that minimizes the intra-class variance, defined as a weighted sum of variances of the two classes:

$$\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t) \qquad (2)$$

Weights ωi are the probabilities that the two classes are separated by a threshold t and σ_i^2 are the variances of these classes. Otsu shows that minimizing the intra-class variance is the same as maximizing inter-class variance:

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = w_1(t)w_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2 \quad (3)$$

which is expressed in terms of class probabilities ωi and class means μi which in turn can be updated iteratively. The class probabilities are estimated as:

$$w_1(t) = \sum_{i=1}^{t} P(i) \qquad w_2(t) = \sum_{i=t+1}^{I} P(i) \qquad (4)$$

And the class means are given by:

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{w_1(t)} \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{w_2(t)} \qquad (5)$$

Finally, the individual class variances are:

$$\sigma_1^2(t) = \sum_{i=1}\left[i - \mu_1(t)\right]^2 \frac{P(i)}{w_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I}\left[i - \mu_2(t)\right]^2 \frac{P(i)}{w_2(t)} \qquad (6)$$

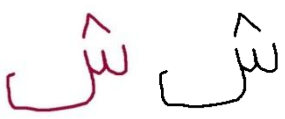
Figure 2. (a) Colored Image (b) binarized image

After we obtain the threshold of the input image, we assign all the pixels values below it to one (white), and all pixel values above it to zeros (black). An example of colored character image that is converted to a black and white image is shown in figure 2.

3. *Resizing*

The last step in the preprocessing phase is to resize the input image to normalize all the input images with different sizes. We perform bicubic interpolation on all the input images. Bicubic produces noticeably sharper images than other interpolation methods such as bilinear and nearest neighbor interpolation, and is perhaps the ideal combination of processing time and output quality. Bicubic Interpolation attempts to reconstruct the exact surface between four initial adjacent pixels. It does this by extracting sixteen pieces of information. Based on the values of the samples, the x slopes of those values, the y slopes of those values, and the xy slope cross products of those values [14], [15], and [16]. It turns out that any point on a two dimensional unity normalized surface can be represented by a set of sixteen cubic polynomial equations. The formulae below give the interpolated value; it is applied to each pixel. The m and n summation span a 4x4 grid around the pixel (i,j).

$$F(i\ ,j) = \sum_{m=-1}^{2}\sum_{n=-1}^{2} F(i+m, j+n)R(m-dx)R(dy-n) \qquad (7)$$

The cubic weighting function R(x) is given below.

$$R(x) = \frac{1}{6}\left[P(x+2)^3 - 4P(x+1)^3 + 6P(x)^3 - 4P(x-1)^3\right]$$

$$P(x) = \begin{cases} x & x > 0 \\ 0 & x \le 0 \end{cases} \qquad (8)$$

Figure 3 shows the original input character image and its resized version

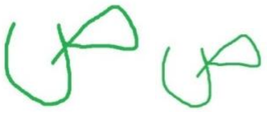
Figure3. (a)Original image (b)resized image

B. *Feature Extraction*

Feature analysis determines the descriptors, or feature set, used to describe all characters. Given a character image, the feature extractor derives the features that the character possesses. The derived features are then used as input to the character classifier [17]. There are 14 features extracted from the character of which 4 are for the whole image as listed below:

- Height / Width
- number of black pixels / number of white pixels image

- number of horizontal transitions
- number of vertical transitions

The horizontal and vertical transition is a technique used to detect the curvature of each character and found to be effective for this purpose. The procedure runs a horizontal scanning through the character box and finds the number of times that the pixel value changes state from 0 to 1 or from 1 to 0 as shown in figure 4. The total number of times that the pixel status changes, is its horizontal transition value. Similar process is used to find the vertical transition value.

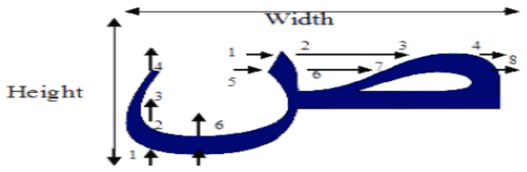

Figure 4. Horizontal and vertical transitions

In addition, the image is divided into four regions as shown in figure 5 and the following features are extracted from these regions:

- Black Pixels in Region 1/White Pixels in Region 1
- Black Pixels in Region 2/White Pixels in Region 2
- Black Pixels in Region 3/White Pixels in Region 3
- Black Pixels in Region 4/White Pixels in Region 4
- Black Pixels in Region 1/Black Pixels in Region 2
- Black Pixels in Region 3/Black Pixels in Region 4
- Black Pixels in Region 1/Black Pixels in Region 3
- Black Pixels in Region 2/Black Pixels in Region 4
- Black Pixels in Region 1/Black Pixels in Region 4
- Black Pixels in Region 2/Black Pixels in Region 3

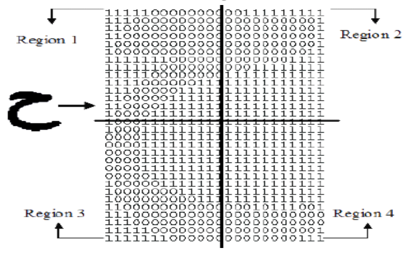

Figure5. Dividing the image to 4 regions and extracting features

These features were found to be sufficient to distinguish between different characters. The extracted feature vector is to train the Neural Network and the k nearest neighbor.

C. Back Propagation Neural Network

Neural network is widely used as a classifier in many handwritten character recognition systems [18]. Also, due to the simplicity, generality, and good learning ability of neural networks, these types of classifiers are found to be more efficient [19]. As we make a lot of computations in the previous two steps, we reduce our problem dimension from the input size (the image size) in the first step, to the number of features extracted in the second step. We can use then only this number of features as input to our classifier.

We design a network and train it to recognize the 28 letters of the Arabic alphabet. The input to this network is the vector containing the 14 features we previously computed, and the target vector is a 28-element vector with a 1 in the position of the letter it represents, and 0's everywhere else. For example, the letter "Alif" is to be represented by a 1 in the first element (as "Alif" is the first letter of the Arabic alphabet), and 0's in elements two through twenty-eight.

The neural network needs 14 inputs and 28 neurons in its output layer to identify the letters. The network is a two-layer log-sigmoid/log-sigmoid network. The log-sigmoid transfer function was picked because its output range (0 to 1) is perfect for learning to output Boolean values. The hidden (first) layer has 10 neurons. This number was chosen by experiment. If the network has trouble learning, then neurons can be added to this layer . After the network is trained the output is passed through the competitive transfer function. This makes sure that the output corresponding to the letter most like the noisy input vector takes on a value of 1, and all others have a value of 0. The result of this post processing is the output that is actually used.

D. K Nearest Neighbor (KNN)

k-Nearest Neighbor is an instance based classification algorithm. Many researchers have found that the kNN algorithm achieves very good performance for character recognition in their experiments on different data sets [20]. The idea behind k-Nearest Neighbor algorithm is quite straightforward. To classify a new character, the system finds the k nearest neighbors among the training datasets, and uses the categories of the k nearest neighbors to weight the category candidates.

The kNN algorithm can be described using the following equation:

$$y(d_i) = \arg\max_k \sum_{x_j \in kNN} Sim(d_i, x_j)\, y(x_j, c_k) \qquad (9)$$

where di is a test character, xj is one of the neighbors in the training set, y(xj,ck) ∈ {0,1} indicates whether xj belongs to class ck, and Sim(di,xj) is the similarity function for di. Equation (9) means the class with maximal sum of similarity will be the winner.

The performance of this algorithm greatly depends on two factors, that is, a suitable similarity function and an appropriate value for the parameter k. The similarity function we use in this research is the Euclidean distance. It is given by equation 10:

$$f(x,p)^2 = \sum_{i=1}^{N} (x_i - p_i)^2 \qquad (10)$$

4. Experimental design

We build our isolated Arabic characters dataset, by asking five volunteer writers to write the 28 letters two times each, each letter was written separately on a white paper, and then we scanned and saved it as a bitmap image. The second time, the writers used the paint software to draw the letters using the mouse or the key pad, and then each letter was saved separately as a bitmap image. At the end of this collecting data phase, we randomly split the dataset into two subsets, 70% for training (196 characters) and 30% for testing (84 characters).

After we collect the data, we implement the whole system using Matlab. The input image is read by Matlab. Then it goes through the preprocessing part, the image is denoised, binarized, and resized to a fixed size (64x64). The preprocessed image enters the second module in our code to extract its main features. We consider this part the most important and critical part in the algorithm because depending on their calculations, the classifiers will be evaluated. And any mistakes in this part will yield to bad results in the classification phase.

The last part is implementing the back propagation neural network and the k nearest neighbor. We used the Neural Network and Bioinformatics toolboxes in the Matlab to code this part. After we built the whole system, we tested the classifiers described in section 3.3 and 3.4 on our constructed database. Our experiments aim to answer the following questions:

1. Are the preprocessing operations important for the classifier to function correctly?
2. Are the extracted features in the second module good enough for the classifiers as character discriminators?
3. Which classifier (BPNN, KNN) is more appropriate for our application in terms of: accuracy, robustness, and efficiency?

5. Results and discussions

A. *The Effect Of The Preprocessing Operations*

First, we should note that binarizing the image is a must, because the feature extraction algorithm depends mainly on the ratio between black pixels and white pixels in the different regions of the image. With converting all images to black and white images, the recognition system becomes more robust and generalized. This is because it can train and test any scanned image with any format and with any color system.

Secondly, there are two reasons after normalizing the images size. First, it will take more time of computations to work on the whole image especially if the input image size is large. Second and more important is that the main idea behind the proposed classifier is to compare the ratio between pixels densities in all images. So, if the images are not of the same size, the ratio of their pixels densities will be meaningless. And the classifier will definitely respond with wrong answers and will not be able to recognize the characters. About the resizing parameters, we experiment the recognition system with different sizes such as 16x16, 32x32, 64x64, and 128x128 (see figure 6). We found that the best performance was with the size 64x64. Smaller sizes reduce the quality of the images, and fine details disappear during the bicubic interpolation process. That leads the classifier to more errors in its recognition to these small images. Also, from our experiments, we found that to enlarge the image more than 64x64 does not enhance the images quality. And consequently doesn't improve the classifier performance.

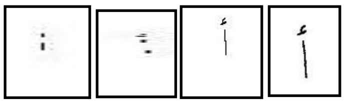
Figure 6. Different sizes for character "Alif"(a)16x16 (b)32x32 (c)64x64 (d) 128x128

Thirdly, the Gaussian filter we use to remove noise has an important parameter, which is sigma. Sigma controls the strength of the filter. When sigma value increases, a blurring effect can be produced. From our experiments, best results are achieved using sigma equal to 0.1.The Gaussian filter also has one more important effect on improving the neural network results; that is smoothing the overall image. The scanned image always contains noise that usually appears as an extra pixel (black or white) in the character image. If the noise is not taken into consideration, it could subvert the process and produce an incorrect result. That is why smoothing the image is important even if no obvious noise is apparent. Comparing the two classifiers we used as shown in figure 7, we find that the performance of the neural network seriously degraded if the input image is not smoothed. On the other hand, we find that k nearest neighbor is very robust against noise. It can correctly classify any character even if heavy noise is added to it. For example, we feed the kNN classifier with letter "Sheen" without filtering it, and kNN recognize it correctly. kNN is robust enough that it didn't confuse between letter "Sheen" and "Seen", where the difference between them is only the stroke.

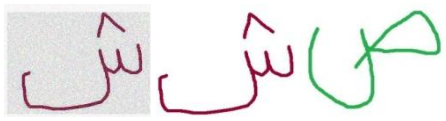
Figure 7.(a)noised image (b)kNN answer (c)BPNN answer

B. *The Effect of Extracted Features*

The features we extract in our research was originally presented in [17], but with different preprocessing techniques and different classifier. In our experiments, we want to investigate the suitability of these features for our suggested classifiers. Inspired with the Sequential Backward Selection (SBS) method, we begin to eliminate one feature after another and then train the classifier using the rest of features, and measure its accuracy.

First, we should note that the first four elements in our feature vector describe the global features of the character. The last ten elements trace the fine details of the character, so that the classifier can discriminate between similar characters. This should be done with care in the Arabic characters because only one dot can transform one character to another. We eliminate one feature after another, and then try to classify the character to test the importance of these features. Using the neural network, we found that the 14 features is a minimal number of features to be used in recognition systems. Every time we eliminate one feature, the classifier performance decreases dramatically. The most effective features are the first four features. This is because they give the global features of the characters. Using these first four features, the classifier can differentiate between the different general categories of the Arabic characters (tall and thin characters, wide characters, highly curved characters, etc.). The other ten features can retrieve the local features of the characters and more detailed classification can be achieved. From our experiments, we find that the 14 featured are essential to recognize the whole 28 isolated Arabic characters using BPNN. We repeat the same experiment using the kNN, and again the kNN proves its robustness. The kNN can still classify the characters correctly after reducing the number of features by 4 (see table 1). This means, kNN can function with only 10 features.

Table 1. Classifiers accuracy with different number of features.

No. of Features	BPNN Accuracy	kNN Accuracy
All 14 F	80(%)	100(%)
From F1 to F10	70(%)	100(%)
From F5 to F14	50(%)	100(%)
From F3 to F12	60(%)	100(%)

C. *The Evaluation of Both Classifiers*

Although it was mentioned consistently in literature that neural network is very suitable technique to learn the handwritten characters, seem to refute this for Arabic characters. In our experiments, we try different architectures for the BPNN. We add hidden layers with different number of nodes. Also, we try different transfer functions. The best results were obtained by the network described in section 3.3, which consists of one hidden layer with 10 nodes. Despite all the training for 5000 epochs and even more during the experiments, the network at its best responds correctly only in 80% of the cases. The neural network performance gets worse when the writer of the characters is different due to differences in their hand style. Also, the neural network takes high computation time because it needs to train for high number of epochs until it converges and its SSE (Sum squared error performance function) is acceptable. For more confirmation, we download different OCR-neural network classifiers programs that work on simpler cases such as recognizing handwritten numbers from 0 to 9. We find that the NN behaves the same and it fails to classify all handwritten numbers.

For our second classifier, the k nearest neighbor, we find that very acceptable results are obtained using the Euclidean distance and by setting k to 1. The kNN responds correctly for all the data belongs to the training datasets, it has 100% accuracy. When it is asked to classify a character never seen before that is written by a different writer, its accuracy decreases to 90% which is still acceptable performance. Moreover, the kNN in test data responds with very similar characters in case it is not the exact one. For example, when it gives wrong answer it classifies letter "Taa" as "Thaa" where the difference between them is just one dot. Another kNN advantages is its speed. The classifier returns the correct label in terms of milliseconds. This is because it does not actually train like neural networks; it computes the distance between the input feature vector and the training samples and responds with the most similar class. A brief comparison between back propagation neural network and k nearest neighbor in recognizing the isolated handwritten Arabic characters is demonstrated in table 2.

Table 2. Comparison between BPNN and KNN

	Training Accuracy (%)	Testing Accuracy (%)	Average SSE	Time in seconds	Sensitive to noise
BPNN	80	60	0.5	13.54	Yes
KNN	100	90	0.0042	0.089	No

6. Conclusions

In this research, we succeed to build a character recognition system for the Arabic characters using two classification algorithms; the BPNN and the kNN. Although we expected that neural network will be the best solution for our problem, we find that kNN performs better. kNN has very low error rate in classifying new datasets, and its accuracy is 100% for the training datasets. Also, kNN is very robust to high noise in the input images. On the contrary, BPNN has high error rate in the training data, and very low accuracy in the testing data. Also, BPNN is sensitive to noise. According to the training time, BPNN needs to train for 5000 epochs before it responds with the character label. On the other hand, kNN is very fast in training the datasets. Using our 2.00 G.Hz processor and 3.0 G.B. RAM computer, kNN responds with the right class 150 times faster than the BPNN for the same input data.

The proposed system can be enhanced by adding further operations in the preprocessing phase so that the letters will be the same size and in the same position, such as scaling, rotation, and translation normalization. We believe that the neural network can behave better and its accuracy increased with these normalized data.

The whole system can be extended to work on the Arabic words instead of the characters by applying segmentation techniques, and recognize the different shapes for each letter instead of working on the isolated form only. That will of course need more research on the appropriate features.

REFERENCES

[1] Abdelmalek Z, ORAN, "A Basis for Arabic OCR system", Proceeding of International Symposium on Intelligent Multimedia, Video and Speech Processing, Hong Kong, pp. 703-706, 2004.

[2] Lorigo, AuthorLiana M., & Govindaraju, Venu, "Offline Arabic Handwriting Recognition: A Survey", Ieee Transactions On Pattern Analysis And Machine Intelligence", 2006.

[3] Tim Klassen, " Towards Neural Network Recognition Of Handwritten Arabic Letters", Thesis, Dalhousie University, 2001.

[4] H. AI-Muallim and S Yamaguchi. "A method of recognition of Arabic cursive handwriting". IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 9, pp. 715-722,1987.

[5] A. Amin and H. Alsadoun, "A new segmentation technique of Arabic text.", IEEE Trans. Pattern Recognition, Vo1.2, pp. 441-445, 1992.

[6] A. Amin and H. Alsadoun, "Hand printed Arabic Character Recognition System", IEEE Trans. Pattern Recognition, Vol. 2, pp536-539, 1994.

[7] S. Abuhaiba, M. 1. 1. Holt, and S. Datta, "Processing of binary images of handwritten text documents," Pattern Recognition, vol. 29, pp. 11611177, 1996.

[8] S. I. Abuhaiba, M. J. 1. Holt, and S. Datta, "Recognition of Off-Line Cursive Handwriting," Computer Vision and Image Understanding, vol. 71, pp. 19-38, 1998.

[9] M. Khorsheed, "Recognising handwritten Arabic manuscripts using a single hidden Markov model", Pattern Recognition Letters, vol. 24, pp. 2235-2242, 2003.

[10] S. Alma'adeed, C. Higgens, and D. Elliman, "Off-line recognition of handwritten Arabic words using multiple hidden Markov models", Knowledge-Based Systems, vol. 17, pp. 75-79, 2004.

[11] S. Alma'adeed, "Recognition of Off-Line Handwritten Arabic Words Using Neural Network", proc. of the Geometric Modeling and Imaging - New Trends, 2006.

[12] Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing", Second Edition, 2002.

[13] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66.

[14]http://local.wasp.uwa.edu.au/~pbourke/texture_colour/imageprocess/

[15] http://www.tinaja.com/glib/pixintpl.pdf

[16] Robert G. Keys: "Cubic convolution interpolation for digital image Processing", IEEE transactions on acoustics, speech, and signal processing, vol. Assp-29, no. 6, December 1981.

[17] Haidar Almohri, John S. Gray, Hisham Alnajjar, " A Real-time DSP-Based Optical Character Recognition System for Isolated Arabic characters using the TI TMS320C6416T", Proceedings of The 2008 IAJC-IJME International Conference, ISBN 978-1-60643-379-9.

[18] K.Khatatneh, "Probabilistic Artificial Neural Network for Recognizing the Arabic. Hand Written Characters", Journal of Computer Science 3 (12), 881-886, 2006.

[19] W. Zhou, "Verification of the Nonparametric Characteristics of Backprobagation Neural Networks for Image Classification",IEEE Transaction on Geoscience and Remote Sensing, vol. 37, no. 2, pp. 771-779, 1999.

[20] Li Baoli1, Yu Shiwen1, and Lu Qin2, "An Improved k-Nearest Neighbor Algorithm for Text Categorization", Proceedings of the 20th International Conference on Computer Processing of Oriental Languages, Shenyang, China, 2003.