International Journal of Imaging,
Autumn 2009, Volume 2, Number A09
ISSN 0974-0627; Copyright © 2009 by IJI, ISDER

Automatic Recognition of Off-line Handwritten Arabic (Indian) Numerals Using Support Vector and Extreme Learning Machines

Sabri A. Mahmoud and **Sunday O. Olatunji**

Information and Computer Science,
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.
Email: {smasaad, olatunji@kfupm.edu.sa}

ABSTRACT

This paper describes a technique using Support Vector (SVM) and Extreme Learning Machines (ELM) for automatic recognition of off-line handwritten Arabic (Indian) numerals. The features of angle, distance, horizontal, and vertical span are extracted from these numerals. The database has 44 writers with 48 samples of each digit totaling 21120 samples. A two-stage exhaustive parameter estimation technique is used to estimate the best values for the SVM parameters for this application. For SVM parameter estimation, the database is split into 4 subsets: three were used in training and validation in turn, and the fourth for testing.

The SVM and ELM classifiers were trained with 75% of the data (i.e. the first 33 writers) and tested with the remaining data (i.e. writers 34 to 44) by using the estimated parameters. The recognition rates of SVM and ELM classifiers at the digit and writers' levels are compared. The training and testing times of SVM and ELM indicate that ELM is much faster to train and test than SVM. The classification errors are analyzed and categorized.

The recognition rates of SVM and ELM classifiers are compared with Hidden Markov Model (HMM) and the Nearest Mean (NM) classifiers. Using the SVM, ELM, HMM and NM classifiers the achieved average recognition rates are 99.39%, 99.45%, 97.99% and 94.35%, respectively. ELM and SVM recognition rates are better for all the digits than HMM and NM.
The presented technique, using simple features and SVM and ELM classifiers, are effective in the recognition of handwritten Arabic (Indian) numerals, and it is shown to be superior to HMM and NM classifiers for all digits.

KeyWords: Arabic (Indian) numeral recognition, OCR, SVM, HMM, Handwritten Digit recognition, Normalization.

2000 Mathematics Subject Classification: 68U10, 94A08, 68T10.

1 Introduction

Handwritten digit recognition is a vital component in many applications, such as office automation, check verification, and a large variety of banking, business, postal address reading, and data entry applications.

Handwritten numeral digits are more difficult to recognize than printed digits, due to the different handwriting styles that are subject to inter- and intra-writer variations. Arabic handwriting has many styles such as Naskh, Kofi, and others. Many writers mix these writing styles [1]. This makes the recognition more difficult, hence requiring more sophisticated feature extraction and recognition techniques.

Arabic Text Recognition (ATR) has not been researched as thoroughly as Latin, Japanese, or Chinese. The lag of research on Arabic may be attributed, in part, to lack of adequate support including lack of bench-marking databases and to the calligraphic nature of the Arabic alphabet as distinguished from other languages.

Indian numerals are used in Arabic writing while Arabic numerals are used in Latin languages. Hence, we will use the term 'Arabic numerals' to refer to the Indian numerals that are used in Arabic writing. Arabic text is cursive while Arabic numerals are not. Figure (1) shows samples of Arabic numerals. Although Arabic text is written from right to left, Arabic numbers are written left to right with most significant digit being the left most one and the least significant digit is the right most one. However, the way the digits are stored in memory is in the reverse order (viz. most significant digit is stored first and so on) which is contrary to the way the number is displayed and seen in the scanned image. For example, the upper number of Figure (1) shows the Arabic (Indian) number 19250. Digit 1 is written first then digits 9, 2, 5, and digit 0 is written last. Digit 1 is the most significant digit and digit 0 is the least. Scanning an Arabic number by scanner and saving the image will give 0 as the right most digit, then 5,2, 9, then 1 as the left most digit.

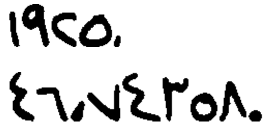


Figure 1 Arabic numerals

A number may consist of an arbitrary number of digits. The recognition system performs classification on each digit independently preserving its relative position with respect to other digits in order to obtain the actual value of the number after recognition. The subsequent stages of the developed recognition system have enough flexibility to treat variations, line thickness, writing size, and translation of the handwritten string. The left-to-right position order of each digit is preserved to account for the digit weight after individual digit classification.

In this paper, we present a simple and effective technique for the recognition of offline handwritten Indian numerals (0,1,..9) used in Arabic writing. The presented technique is implemented using SVM and ELM classifiers. A two-stage parameter estimation technique (coarse- and fine-search) is implemented to estimate the values of the SVM parameters that produce the highest recognition rates. The estimated parameters are used in the recognition of Arabic digits using SVM. The recognition rates of SVM and ELM classifiers are compared at the digit and writers' levels. The results of SVM and ELM are also compared with published work using the same data and features with HMM and the Nearest Mean (NM) classifiers. The results of the four classifiers are analyzed and compared. The recognition rates of SVM and ELM are superior to those of HMM and MN classifiers for all the digits as shown in Section 6.

This paper is organized as follows:

- The literature review is presented in Section 2.
- Feature extraction is addressed in Section 3 where four types of features are used.

- SVM theory is presented in Section 4.
- Section 5 addresses ELM theory.
- Training, recognition, and experimental results are addressed in Section 6.
- The conclusions are presented in Section 7.

2 Literature review

Various methods have been proposed and high recognition rates are reported for the recognition of English handwritten digits [2-6]. In recent years many researchers addressed the recognition of Arabic text including Arabic numerals [7-15]. Surveys on Arabic Optical Text Recognition may be cited in [16, 17] and a bibliography in [18]. Bazzi et al presented a system for bilingual text recognition (English/Arabic) [19, 20]. In addition, several researchers reported the recognition of Persian /Arabic handwritten digits [21-25]. However, the reported recognition rates need more improvements to be practical.

Al-Omari presented a recognition system for Indian numeral digits using average template matching approaches [7]. Freehand sketches of online numeric digits placed on an image template were processed to extract a key feature vector representing significant boundary point distances from the digit center of gravity (COG). A model for each digit is formed by processing 30 handwritten digit samples. Classification was made using the Euclidean distance between the feature vector of the test samples and the models. In another study, Al-Omari et al presented a recognition system for online handwritten Indian numerals one to nine. The system skeletonizes the digits and then the geometrical features of the skeleton of the digits are extracted. Probabilistic neural networks (PNNs) are used for classification. The developed system is translation-, rotation-, and scaling-invariant. The authors claim that the system may be extended to address Arabic characters [8].

Bousalma [9] presented an algorithm based on structural techniques for extracting local features from the geometric and topological properties of online Arabic characters by using fuzzy logic. Salah et al. developed a serial model for visual digit classification based on primitive selective attention mechanism [10]. The technique was based on parallel scanning of a down sampled image to find interesting locations through a saliency map, and by extracting key features at those locations at high resolution.

Shahrezea et al. used the shadow coding method for recognition of Persian handwritten digits. In this method, a segment mask is overlaid on the digit image and the features are calculated by projecting the image pixels into these segments [21]. In [22] the Persian digit images are represented by line segments which are used to model and recognize the digits. Additional features and classifiers are needed for discriminating the digit pairs ‘‘0–5’’, ‘‘7–8’’, ‘‘4–6’’. Said et al. fed the pixels of the normalized digit image into a neural network for classification, where the number of the hidden units for the neural network classifier is determined dynamically [23]. Sadri et al. used a feature vector of length 16 which is estimated from the derivative of the horizontal and vertical profiles of the image [24].

Soltanzadeh [25] used the normalized image profile, which is calculated at multiple orientations, as the main feature for the recognition of Persian handwritten digits. The crossing counts and projection histogram, calculated at multiple orientations are used as complementary features. The authors indicated that most of the system errors occurred in discriminating the digits ‘‘2’’, ‘‘3’’, ‘‘4’’ and ‘‘0’’, ‘‘5’’. Hence, discriminating these digits requires the use of additional features and may require the use of additional classifiers.

Support Vector Machines (SVMs) are modern learning systems that deliver state-of-the-art performance in real world pattern recognition and data-mining applications such as text categorization, hand-written character recognition, image classification and bioinformatics. The SVM was developed by Vapnik and other researchers [26-30]. Within a short period of time SVM classifiers became competitive with the best available systems for pattern recognition applications [31, 32]. Camastra [33] used SVM for cursive character recognition. Mozaffari et al used SVM and direction and curvature features of the skeleton images to recognize Arabic/Persian zip code numerals [34]. Soltanzadeh used the profile of the digit images along with SVM to recognize Arabic/Persian digits [35], whereas Mozaffari et al used the SVM classifier to perform feature comparison between fractal codes and wavelet transform [36]. Mowlaei et al [37] used wavelet transforms with SVM to recognize Arabic/Persian digits extracted from postal addresses.

3 Feature Extraction

In this work, we are using four types of features. Angular, distance, horizontal and vertical span features. The following subsections address these features.

3.1 Angle Span Features

The digit center of gravity (COG) is used as the center point for calculating this feature to make it translation invariant. The image of the Arabic numeral is sliced using angular lines with angles of α degrees between consecutive lines passing through the COG. If a black pixel falls in a slice, then the counter of the slice is incremented. This procedure is repeated for other slices. Altogether $360/\alpha$ slices (features) are extracted. These features are normalized by dividing the number of black pixels in each slice by the total number of black pixels of the digit.

3.2 Distance Span Features

The Euclidean distance d between COG and digit image origin (which is the top-most and left-most pixel) is calculated. Several concentric circles having centre at COG are used. The distance, d, is divided into S divisions each division div=d/S, where S is the number of concentric circles used. The first circle radius r_1 = div. The second circle radius is r_2 = 2*div. Other circles radiuses are r_i = i*div.

Let the number of black pixels in each concentric circuit, i, be P_i. The first feature value, f_1, is equal to P_1. Other features are computed from the number of pixels P_i in each circle. $f_i = P_i - P_{i-1}$, where i = 2 ... S. The first feature is equal to P_1, the second feature is $P_2 - P_1$, and so on. The remaining black pixels that do not fall in any of the concentric circles are used as the last feature. These features are then normalized by dividing them by the total number of pixels of the Arabic numeral.

3.3 Horizontal and vertical span features

The whole image is divided into a number of equal horizontal and vertical bars. The number of bars in each dimension is taken, in our case, as 20 in the horizontal and 20 in the vertical directions. The horizontal span (height of the digit) and the vertical span (the width of the digit) features are normalized as some numerals are small (e.g. the zero) while other numerals are large (e.g. the digits 4 and 9). For each digit, the number of black pixels in the horizontal and vertical bars is calculated. To normalize these features we divide them by the total number of black pixels in the image.

The details for calculating the above four types of features are found in [1]. In our present paper, we will use the same number of features as in [1] for comparison purposes. Hence, we will slice the character at 5^o (i.e. $\alpha = 5^o$), by using 7 slicing circles and taking 20 horizontal and 20 vertical segments (a total of 120 features). This number of features gave the best recognition rates with HMM and NM classifiers as detailed in [1].

4 Support Vector Machines (SVMs)

(SVMs are modern learning systems that deliver state-of-the-art performance in real world pattern recognition and data-mining applications such as text categorization, hand-written character recognition, image classification and bioinformatics.

Here we briefly describe the basic theory behind SVM for pattern recognition, especially for the two-class classification problem, and we refer readers to [27, 38], for a full description of the technique.

For a two-class classification problem, assume that we have a set of samples, i.e. a series of input vectors: $x \in R^d \quad (i = 1, 2, \ldots.N)$ with corresponding labels, $y_i \in \{+1, -1\} \, (i = 1, 2, \ldots.N)$. Here, +1 and −1 indicate the two classes.

The goal is to construct a binary classifier from the available samples which has a small probability of misclassifying a future sample. SVM maps the input vectors $x \in R^d$ into a high dimensional feature space $\Phi(x) \in \mathrm{H}$ and it constructs an Optimal Separating Hyper-plane (OSH). OSH maximizes the margin, the distance between the hyper plane and the nearest data points of each class in the space H . Different mappings construct different SVMs. The mapping $\Phi(\cdot)$ is performed by a kernel function $K(x_i, x_j)$ which defines an inner product in the space H . The decision function implemented by SVM can be written as:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{N} y_i \alpha_i \cdot K(x, x_i) + b \right) \qquad (1)$$

where the coefficients α_i are obtained by solving the following convex Quadratic Programming (QP) problem:

Maximize $$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \cdot y_i y_j \cdot K(x_i, x_j) \, , \; 0 \le \alpha_i \le C \quad (2)$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \qquad i = 1, 2, \ldots..N. \qquad (3)$$

C is a regularization parameter which controls the trade-off between margin and misclassification error. These x_j are called Support Vectors only if the corresponding $\alpha_i > 0$.

Several typical kernel functions are:

Polynomial: $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$ (4)

Radial Basis Function (RBF): $K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^d\right)$ (5)

Linear: $K(x_i, x_j) = x_i^T x_j$ (6)

Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$. (7)

Here $\gamma, r, and \quad d$ are kernel parameters.

SVMs provide a new approach to the problem of pattern recognition. They differ radically from comparable approaches such as neural networks. SVM training always finds a global minimum. An SVM is largely characterized by the choice of its kernel. SVMs thus link the problems they are designed for with a large body of existing work on kernel-based methods.

5 Extreme Learning Machines

In general, the learning rate of Feed-Forward Neural Networks (FFNN) is time-consuming. This limits the scalability of FFNN. There are two main reasons behind this behavior. One is the slow gradient-based learning algorithms used to train the neural network (NN) and the other is the iterative tuning of the parameters of the networks by these learning algorithms Huang et al. [39]. Huang et al. Extreme learning machines (ELM) as proposed by Huang et al. [39] for Single hidden Layer Feed-forward Neural networks (SLFNs) randomly select the input weights and analytically determine the output weights of SLFNs. Huang et al. claim that, in theory, this algorithm tends to provide the best generalization performance at an extremely fast learning speed.

The ELM has several interesting and significant features different from traditional popular gradient-based learning algorithms for feed-forward neural networks. The learning speed of ELM is extremely fast. Huang et al. [40] reported that the simulations show that the learning phase of ELM can be completed in seconds or less than seconds for many applications. In most cases the ELM has better generalization performance than the gradient-based learning such as back propagation. Unlike the traditional classic gradient-based learning algorithms which only work for differentiable activation functions, the ELM learning algorithm can be used to train SLFNs with many non-differentiable activation functions, Huang et. al. [41].

The standard Single-hidden Layer Feed-forward Neural networks (SLFN) with $\tilde{N}$ hidden neurons and activation function g(x) is defended as:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i . x_j + b_i) = o_j, j = 1, \ldots, N ,$$

Where N is the number of samples,

$x_j = [x_{j1}, x_{j2}, \dots , x_{jn}]^T \in R^n$, is the input and n is the number of the input features;

$w_i = [w_{i1}, w_{i2}, \dots , w_{in}]^T$ is the weight vector that connects the *i*th hidden neuron with the input;

$\beta_i = [\beta_{i1}, \beta_{i2}, \dots , \beta_{im}]^T$ is the weight vector that connects the *i*th hidden neuron and the output neurons and b_i is the threshold of the *i*th hidden neuron, m is the number of the outputs, o_j is the corresponding output for data input x_j and the "." in $w_i \,.\, x_j$ means the inner product of w_i and x_j.

The samples, x_i, are presented with their truth values (the class corresponding to the features), t_i, to SLFN.

SLFN aims to minimize the difference between the output, o_j , and the expected output, t_j.

The ELM training algorithm given in Huang et al. [39] works as follows given a training set $N = \{(x_i, t_i) \mid x_i \in \mathbf{R}^n, t_i \in \mathbf{R}^m, i = 1, \dots, N\}$ with activation function given by g(x) and $\tilde{N}$ hidden neurons:

1. Assign random value to the input weights w_i and the bias b_i, $i = 1, \dots , \tilde{N}$
2. Calculate the hidden layer output matrix H.
3. Calculate the output weight β: H β = T, where

$$\beta = \begin{bmatrix} \beta_1^T \\ .. \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m}, \qquad T = \begin{bmatrix} t_1^T \\ .. \\ t_{\tilde{N}}^T \end{bmatrix}_{N \times m}, \quad \text{and}$$

$$\mathbf{H}(w_1, .., w_{\tilde{N}}, b_1, .., b_{\tilde{N}}, x_1, .., x_N) = \begin{bmatrix} g(w_1.x_1 + b_1) & .. & g(w_{\tilde{N}}.x_{\tilde{N}} + b_{\tilde{N}}) \\ . & & . \\ & .. & \\ . & & . \\ g(w_1.x_N + b_1) & .. & g(w_{\tilde{N}}.x_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}$$

In the training phase, the ideal case is training the N samples with zero errors which corresponds to $\sum_{i=1}^{\tilde{N}} \beta_i g(w_i . x_j + b_i) = t_j, j = 1, \dots, N$, and hence, $\sum_{j=1}^{N} \| o_j - t_j \| = 0$.

The solution is $\hat{\beta} = H^{\dagger} T$, where $H^{\dagger}$ is called the Moore-Perrose generalized inverse [39]. The most important properties of this solution as claimed by the authors of [39] are:

1- Minimum training errors

2- Smallest norm of weights and best generalization performance

3- The minimum norm least square solution of Hβ = T is unique, which is $\hat{\beta} = H^{\dagger} T$.

Figure 2 shows an ELM network with inputs (corresponding to the features), hidden neurons, and outputs (corresponding to classes). X_f is feature f, f is the number of input features, O_C is output C and C is the number of outputs.

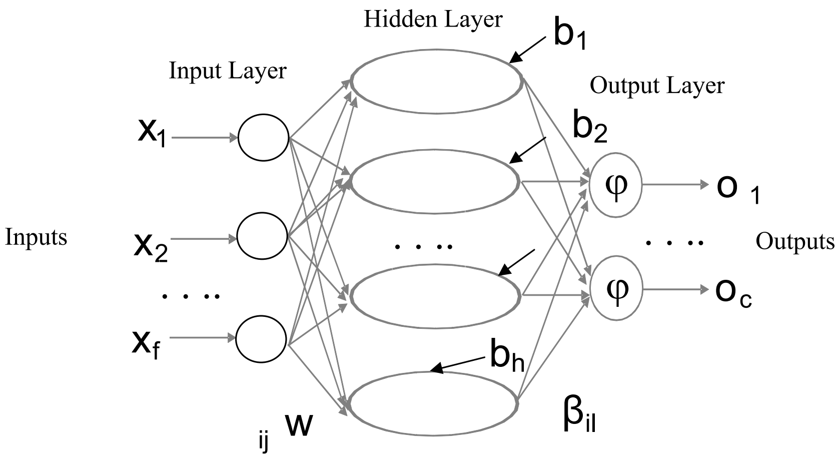


Figure 2 ELM model with input, hidden, and output layers.

5. Training and Recognition

There is no generally accepted database for Arabic numeral recognition that is freely available for researchers. Different researchers of Arabic numeral recognition use different data, and hence the recognition rates of the different techniques may not be comparable. In our present study, we use the data that is freely available on-line [42] and reported in [1]. It consists of a database written by 44 writers consisting of 21120 samples of digits. Figure 3 shows samples of the database digits.

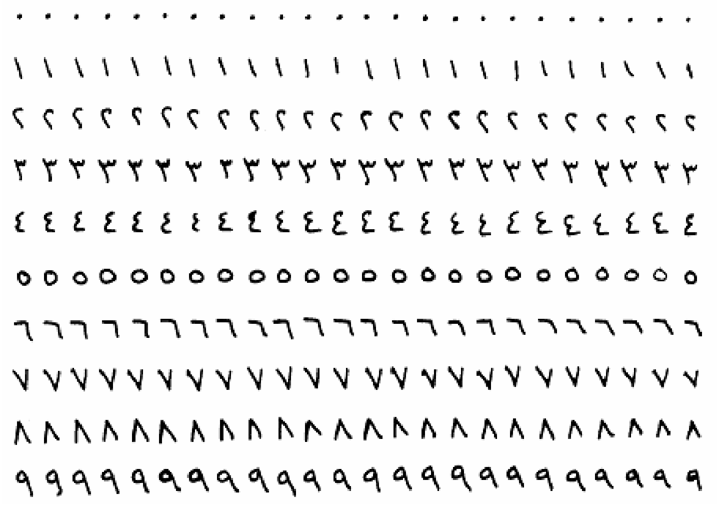


Figure 3 Samples of handwritten Arabic (Indian) digits of the used database.

In our present paper, the SVM and ELM classifiers are applied. Initially a parameter tuning phase is used to estimate the parameters that give the highest recognition rate. An exhaustive search algorithm is carried out to estimate the suitable SVM parameters in two stages. In the first stage, coarse-search parameters estimation is implemented and the best parameters are selected for the second fine-search parameter estimation. These parameters are then used in experimenting with SVM by using the extracted features. We experiment with SVM and ELM classifiers, and then we compared the recognition rates of SVM and ELM with the results of the published work using the same features and HMM and NM classifiers [1].

In general, in the training phase, the features of training data are computed and saved as models for the trained classes. In the recognition phase, an unknown character features are extracted and compared with the features of the models. The unknown character is assigned to the class of the model whose features are the closest (or the most probable) to the features of the unknown character. The implementation of this work is done by using C language and MATLAB. The HTK tools [43] are used in the experimentation of HMM.

6.1 SVM parameter tuning

There are two parameters of SVM using RBF kernel, C and γ. It is not known a priori which parameter values of C and γ is best for our problem. Hence, we experimented with different values of C and γ and selected the best performing ones. The technique of Hsu [44] is implemented with modifications to suit our problem.

The handwritten database which consists of digits written by 44 writers is divided into four subsets. Set S_1 consists of the data written by the first 11 writers (i.e. writers 1 to 11). Sets S_2, S_3, S_4 consist of data written by writers 12 to 22, writers 23 to 33, and writers 34 to 44, respectively. Subsets S_1, S_2, S_3 are used for training and validation in turn, and set S_4 is used for testing. Two sets of S_1 - S_3 are used for training, and the third for validation. For example, sets S_1 and S_2 are used for training and set S_3 for validation, and then sets S_2 and S_3 are used for training and set S_1 for validation and so on. The results of validation are used to select the parameters that have the highest average recognition rate by using the validation set. These selected parameters are used in the testing phase by using the test set, S_4. This process is repeated twice, once with coarse grid-search and the second with fine grid-search. The motivation for this grid-search approach is two-fold. One reason is that we may not feel safe in using methods which avoid an exhaustive parameter search by approximations or heuristics. The second reason is that the computational time to find good parameters by grid-search is not much greater than that by advanced methods, since there are only two parameters [44].

We conducted a coarse grid-search by using parameters of {C = 2^{-5}, 2^{-3}, 2^{-5}, 2^{-1}, 2^{1}, 2^{3}, …., 2^{13}, 2^{15}, and γ = 2^{-15}, 2^{-13}, 2^{-11},…., 2^{-1}, 2^{1}, 2^{3} }. The average recognition rate with the rotating training and validation data subsets is 87.22%, with a maximum of 98.28%, a minimum of 42.39%, and standard deviation of 16.73. The minimum is at C=2^{-5} and $\gamma = 2^{-13}$. The maxima were at different values of C and γ, for example, (C=2^{13} and $\gamma = 2^{-13}$) and (C=2^{15} and $\gamma = 2^{-13}$). It is to be noted that many other values of C and γ are close to the maximum and minimum.

Table 1 shows the recognition rates that are above 98% of the different subsets with different values of C and γ. Other values were not shown, as the table would be difficult to present and we are not interested in the values of C and γ with low recognition rates. Column R_1 shows the recognition rates with subsets S_2 and S_3 in training and S_1 in validation. Column R_2 shows the recognition rates with subsets S_1 and S_3 in training and S_2 in validation. Column R_3 shows the recognition rates with subsets S_1 and S_2 in training and S_3 in validation. Column R_{avg} shows the average recognition rates of columns R_1, R_2, and R_3.

Table 1 Recognition rates with subsets S_1, S_2, S_3 in training (2 subsets) and validation (the third subset) in turn and their average.

		R_1	R_2	R_3	R_{avg}
C	γ	%	%	%	%
2^7	2^-15	96.99	98.62	99.05	98.22
2^7	2^-13	96.99	98.64	99.07	98.23
2^7	2^-11	97.03	98.67	99.07	98.26
2^7	2^-9	96.89	98.66	99.05	98.20
2^7	2^-7	96.7	98.67	99	98.12
2^9	2^-15	96.7	98.6	99.3	98.20
2^9	2^-13	96.72	98.62	99.28	98.21
2^9	2^-11	96.76	98.64	99.28	98.23
2^9	2^-9	96.78	98.73	99.15	98.22
2^9	2^-7	96.7	98.69	99.03	98.14
2^11	2^-15	96.67	98.62	99.43	98.24
2^11	2^-13	96.7	98.64	99.41	98.25
2^11	2^-11	96.74	98.64	99.43	98.27
2^11	2^-9	96.78	98.73	99.17	98.23
2^11	2^-7	96.7	98.69	99.03	98.14
2^13	2^-15	96.67	98.62	99.53	98.27
2^13	2^-13	96.7	98.64	99.49	98.28
2^13	2^-11	96.74	98.64	99.43	98.27
2^13	2^-9	96.78	98.73	99.17	98.23
2^13	2^-7	96.7	98.69	99.03	98.14
2^15	2^-15	96.67	98.62	99.47	98.25
2^15	2^-13	96.7	98.64	99.49	98.28
2^15	2^-11	96.74	98.64	99.43	98.27
2^15	2^-9	96.78	98.73	99.17	98.23
2^15	2^-7	96.7	98.69	99.03	98.14

The above phase was followed by a fine grid-search with parameters of C from 2^7 to 2^{15} in steps of 0.25 and γ from 2^{-16} to 2^{-9} in steps of 0.25. The average recognition rate with the rotating training and validation data subsets is 98.28%, with a maximum of 98.31%, a minimum of 98.23%, and standard deviation of 0.014. The minimum is at $C=2^{-5}$ and $\gamma = 2^{-13}$. The maxima were at different values of C and γ, for example, ($C=2^{13}$ and $\gamma = 2^{-13}$) and ($C=2^{15}$ and $\gamma = 2^{-13}$). It is to be noted that many other values of C and γ are close to the maximum and minimum. Figure 4 shows range of values of C and γ that have average recognition rates at maxima or close to maxima with the training/validation sets. Other values of C and γ have average recognition rates close to maxima. These results show that several values of C and γ produce high recognition rates, and not only one specific value of C and γ. The difference in the average recognition rates between the maxima and minima is below 0.1% for the range of values of C and γ. This indicates that SVM is not very sensitive to these parameters.

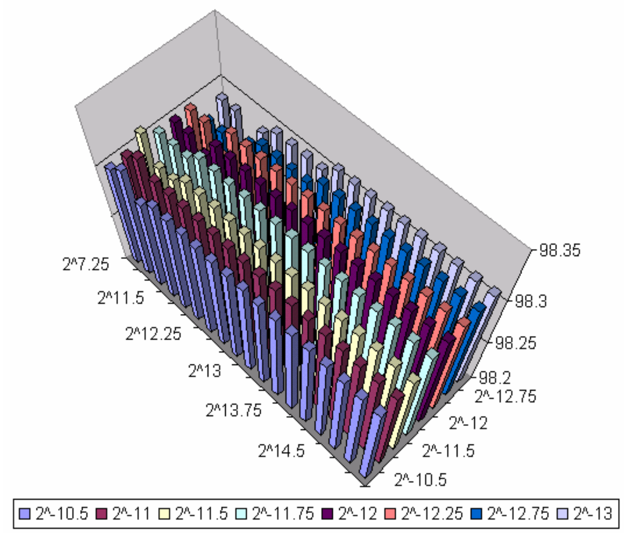


Figure 4 Average recognition rates with the selected C and γ parameters of the fine parameter search with the validation sets as specified with the course parameter estimation of Table (1).

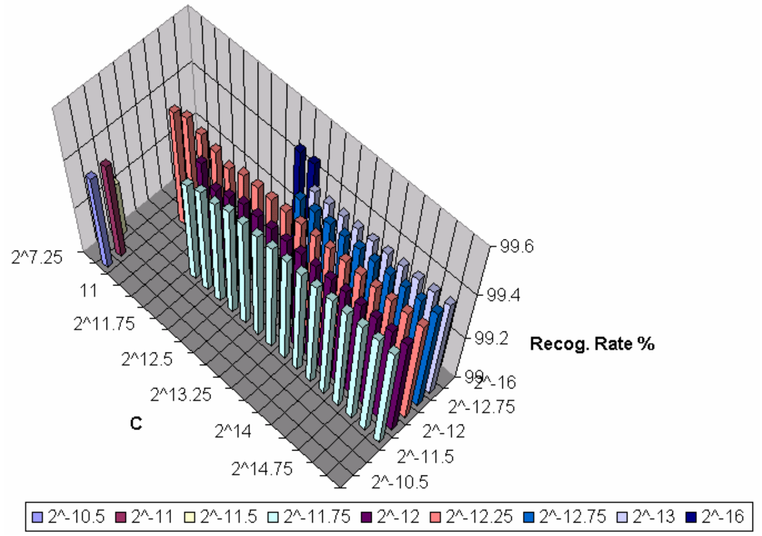


Figure 5 Recognition rates with the selected C and γ parameters of the fine parameter search with all the training data in training the SVM classifier and with the test set used in testing it. Only recognition rates above or equal to 99.39% are shown.

The values of the C and γ parameters that gave best recognition rates in Figure 5 are used in training the SVM classifier with all the training data (i.e. S_1, S_2, and S_3) and testing the SVM classifier with the test data (i.e. S_4, digits written by writers 34 to 44). The recognition rates equal to or greater than 99.39% are shown in Figure 4 with corresponding values of C and γ parameters. The maximum recognition rate is 99.49%, the minimum is 99.22%, the average is 99.44%, and the standard deviation is 0.031.

6.2 SVM and ELM experimentations

As can be seen from the results of the previous section, many values of C and γ produce high recognition rates with the RBF. We experimented with different parameter values. We trained SVM by using the first three subsets (i.e. S_1, S_2, S_3 which include the data written by the first 33 writers) in training and subset S_4 (i.e. writers 34 to 44) was used in testing. The confusion matrix of the results with $C = 2^{12}$ and $\gamma = 2^{-12.25}$ is shown in Table 2. The average recognition rate is 99.39% and the digits recognition rate range between 98.11 in the case of digit 3 and 100% in the case of 1.

Table 2 SVM confusion matrix using the first 75% of the data in training and the remaining 25% of the data in testing with $C = 2^{12}$ and $\gamma = 2^{-12.25}$

	0	1	2	3	4	5	6	7	8	9	%c	%e
0	527	0	1	0	0	0	0	0	0	0	99.81	0.19
1	0	528	0	0	0	0	0	0	0	0	100.00	0.00
2	0	0	521	0	7	0	0	0	0	0	98.67	1.33
3	0	0	0	518	2	0	0	3	1	4	98.11	1.89
4	0	2	2	0	524	0	0	0	0	0	99.24	0.76
5	2	0	0	0	0	526	0	0	0	0	99.62	0.38
6	1	1	0	0	0	0	526	0	0	0	99.62	0.38
7	1	0	0	0	0	0	0	527	0	0	99.81	0.19
8	0	0	0	0	0	0	0	0	526	2	99.62	0.38
9	0	1	0	0	1	0	0	0	0	525	99.43	0.57
											99.39	0.61

Similarly, ELM classifier was trained by using the first 75% of the data and tested by using the remaining 25% of the data. The sigmoid activation function was used in the hidden neurons and the outputs were taken linear. The optimal number of hidden nodes was estimated by using training-validation technique in which the number of hidden nodes that resulted in the highest recognition rate (1000 hidden nodes) was used. We used a vector of 120 inputs corresponding to the features and ten outputs corresponding to digits. The confusion matrix of the recognition results is given by Table 3. The two confusion matrices of SVM and ELM show that the recognition rates of the classifiers are comparable.

Table 3 ELM confusion matrix with the first 75% of the data used in training and the remaining 25% of the data in testing.

	0	1	2	3	4	5	6	7	8	9	%c	%e
0	527	1	0	0	0	0	0	0	0	0	99.81	0.19
1	0	528	0	0	0	0	0	0	0	0	100.00	0.00
2	0	0	526	0	1	0	0	0	0	1	99.62	0.38
3	1	0	1	515	2	0	0	3	0	6	97.54	2.46
4	0	2	1	0	524	0	0	0	0	1	99.24	0.76
5	1	0	0	0	0	525	0	0	2	0	99.43	0.57
6	0	1	0	0	0	0	527	0	0	0	99.81	0.19
7	0	0	0	0	0	0	0	526	0	2	99.62	0.38
8	0	0	0	0	0	0	0	0	527	1	99.81	0.19
9	0	1	0	0	1	0	0	0	0	526	99.62	0.38
											99.45	0.55

Figure 6 shows the recognition rates of the SVM and ELM classifiers for the tested writers (i.e. 34 to 44). Both classifiers were trained with the data of writers 1 to 33 and tested with the data of the remaining writers 34 to 44. The figure shows that the recognition rates of the two classifiers are comparable. SVM has better recognition rates for some writers, but ELM has better rates for other writers. In general, they have similar recognition rates.

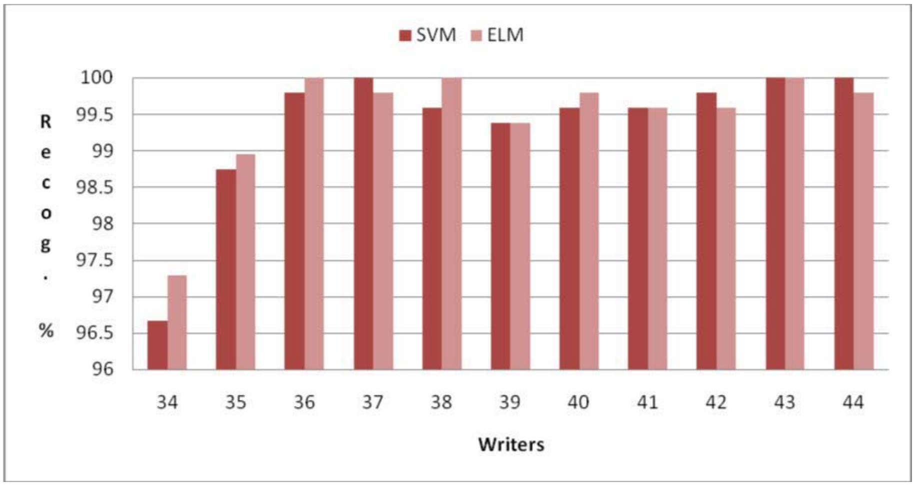


Figure 6 Recognition rates of the SVM and, ELM classifiers which were trained with the data of the first 33 writers and tested with the remaining writers (i.e. 34 to 44).

The training and testing times of the SVM and ELM classifiers were estimated by using a Pentium PC with 2.6 GHz processor and 504 MB RAM. The overall training times were 1267.19 and 145.14 seconds for the SVM and ELM classifiers, respectively. The overall testing times were 7.031, 2.296 seconds for the SVM, ELM, respectively. The testing times of the SVM and ELM for the tested writers 34 to 44 are shown in Figure 7. It is clear that ELM classifier is 8 times faster to train and 3 times faster to test than the SVM classifiers.

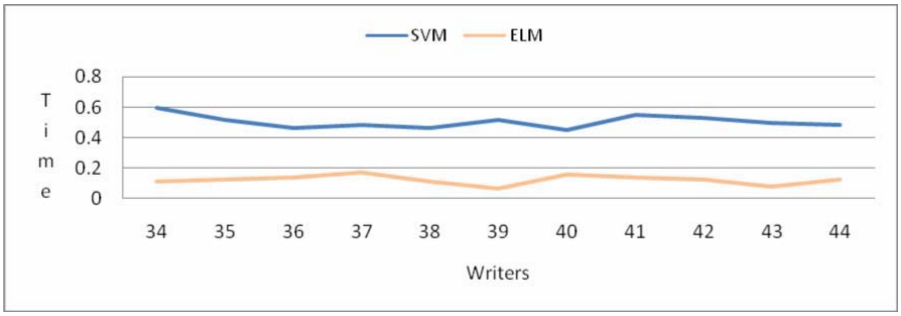


Figure 7 Testing times of the SVM and, ELM classifiers for writers 34 to 44.

6.3 SVM, ELM, HMM, NM classifiers

In this section, we are comparing the results of the SVM and ELM with those of published work which used HMM and NM classifiers and the same features, training and testing data [1].

The HMM results were produced by using a left-to-right HMM which is in line with several research works using HMM [19, 20]. The used model allows relatively large variations in the horizontal position of the Arabic numeral. The sequence of state transition in the training and testing of the model is related to each digit feature observations. The same number of states for all digits was used as in [19, 20]. Each Arabic numeral is represented by 120-dimensional feature vector (viz. 72 angle, 8 distance, 20 horizontal, and 20 vertical span features). We divided our feature vector to twelve separate sub-vectors of 10 features each. Hence, each digit is represented by 12 observations of 10 features each. Approximately 75% of the data was used in training HMM and the remaining 25% was used in testing.

In the case of the NM Classifier, the feature vectors (*V*) of the training data, consisting of the digits written by the first 33 writers, were extracted (viz. 72 Angle, 8 distance, 20 horizontal and 20 vertical span features) in the training phase. The features of each digit are averaged, and the averaged features are used as the models of the Arabic numerals. In the testing phase, using the remaining digits written by 11 writers, the feature vector (*V*) for the unknown character is computed and then compared to the feature vectors of the model classes. The classification decision is based on the nearest mean neighbor classification method. The nearest distance is computed using a simple formula given by $E_i = \sum_{j=1}^{k} \left| M_{ij} - V_j \right|$, where E_i is the distance between the input digit and model *i* (i.e. sum of the absolute differences between the features of the input digit and those of model *i*), *k* is the total number of parameters in the feature vector (i.e. 120), M_{ij} is the j^{th} feature of model *i*, and V_j is feature *j* of the test digit feature vector.

The distances (E_i) between the new digit and all models' feature vectors are found. The argument of the minimum value found (i.e. min (E_i)) yields the recognized model *i.* This model is considered as the class that matches most closely the obtained features vector of the unknown digit. Hence, the digit is labeled with that class.

Figure 8 shows the recognition rates of the different digits with SVM, ELM, HMM, and NM. It is clear from the figure that SVM and ELM outperforms the HMM and NM classifiers. On average the SVM and ELM recognition rates are nearly 1.4% greater than HMM and 5% greater than those of NM classifiers. In some cases, the difference is 2.1% with HMM (as is the case with digit 9) and 10% with NM (as is the case with digit 5).

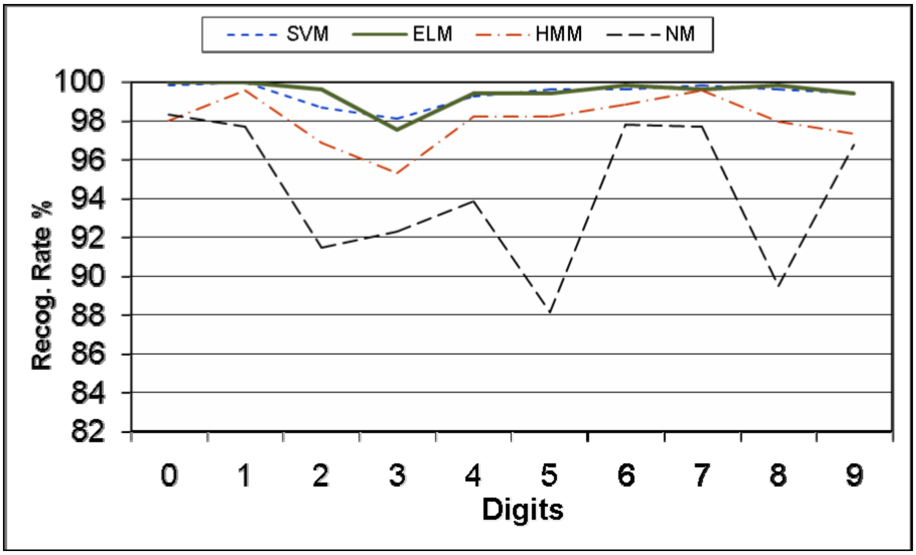


Figure 8 Recognition rates of the SVM, ELM, HMM and NM classifiers.

Figure 9 shows all the digits that were misclassified by SVM and ELM classifiers. The figure shows the tested digit, its image, recognized as, and the writer. In the column recognized as the recognized digit and the classifier that misrecognized the digit is given. S is used for SVM, E for ELM, and B when both classifiers misrecognized the digit. Each row consists of four samples. Analyzing the misclassified digits the reasons for the misclassification errors may be attributed to the following main categories:

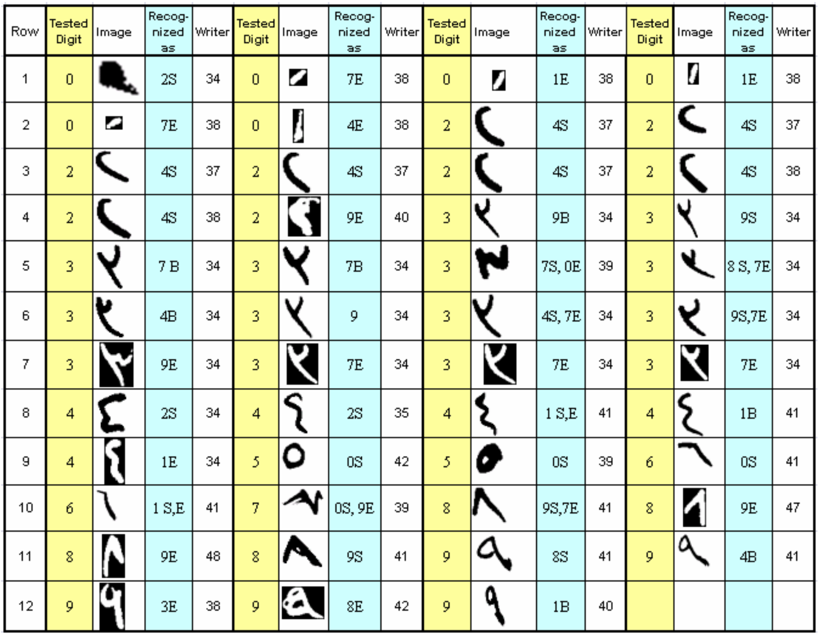
1. Errors due to bad or corrupted data. The samples that fall under this category are: 1st sample of 1st row; 2nd sample of 2nd row; samples 3 of 3th row; 2nd of 4th row; 3rd samples of 5th row; 1st samples of 7th row; 1st , 2nd, 3rd samples of the 8th row; 1st and 3rd samples of the 9th row; 1st and 2nd samples of the 10th row; 4th sample of the 11th row and 2nd sample of the 12th row.. This accounts for 32% of the errors. These samples may not be recognized by human without context. For example, the first sample is a dot and looks as small filled digit 2 (was recognized as 2). 3nd Sample of the 3rd row is not a four as recognized (a person may confuse it with a parenthesis). Referring to the samples of 5th row, a human may not be sure what is the 3nd sample if given without context; the 1st sample of the 8th row could be confused between a 4 or a 2 (it was recognized as 2.

2. Errors due to deformed samples or samples with un-proportional segments in relation to other segments in length and orientation or skewed samples. The samples that fall under this category

are the 4th sample of 1st row; 3th sample of 2nd row; 1th , 2nd, 4th samples of 3rd row; 1st sample of 4th row; 1st sample of 6th row, These samples account for 15% of the misclassification errors.

3. Genuine errors that are misclassified with no visible reason and which can be attributed to insufficient classification capability of the used features and classifier. The remaining samples in Figure (8) fall under this category. These samples account for 53% of the errors (or 0.3% of the overall recognition rate). It may not be expected to achieve 100% recognition rate for writer-independent handwritten offline digits from any classifier.

4. Errors of digits 2 and 3 account for 47% of the errors. Errors of digits written by writers 34, 38, 41 account for 34%, 17%, 17% respectively (a total of 68% of the errors are of digits written be these writers).



Row	Tested Digit	Image	Recog-nized as	Writer	Tested Digit	Image	Recog-nized as	Writer	Tested Digit	Image	Recog-nized as	Writer	Tested Digit	Image	Recog-nized as	Writer
1	0		2S	34	0		7E	38	0		1E	38	0		1E	38
2	0		7E	38	0		4E	38	2		4S	37	2		4S	37
3	2		4S	37	2		4S	37	2		4S	37	2		4S	38
4	2		4S	38	2		9E	40	3		9B	34	3		9S	34
5	3		7 B	34	3		7B	34	3		7S, 0E	39	3		8 S, 7E	34
6	3		4B	34	3		9	34	3		4S, 7E	34	3		9S,7E	34
7	3		9E	34	3		7E	34	3		7E	34	3		7E	34
8	4		2S	34	4		2S	35	4		1 S,E	41	4		1B	41
9	4		1E	34	5		0S	42	5		0S	39	6		0S	41
10	6		1 S,E	41	7		0S, 9E	39	8		9S,7E	41	8		9E	47
11	8		9E	48	8		9S	41	9		8S	41	9		4B	41
12	9		3E	38	9		8E	42	9		1B	40				

Figure 9 Misclassified digits.

Figure 10 shows samples of digits that were misclassified using HMM and NM classifiers and they were correctly classified using SVM classifier. Column 1 gives the sample truth value. As can be seen from the figure, there are samples of the above first and second categories that were correctly classified by SVM and ELM, which indicates the robustness of the technique for digit deformations.

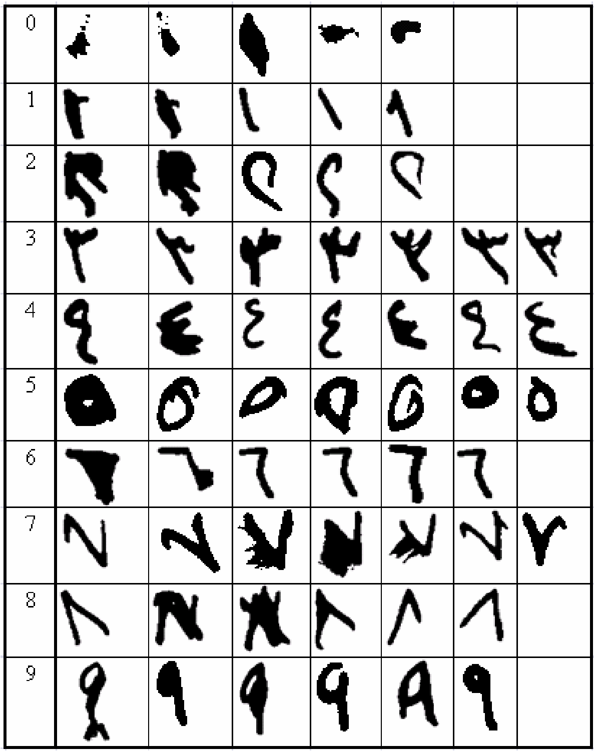


Figure 10 Samples of digits that were misclassified by HMM and NM classifiers and were correctly classified by SVM and ELM classifiers.

7 Conclusions

This paper presents a system handwritten Arabic numeral recognition based on estimating simple and effective features (viz. angular, distance, horizontal, and vertical span features). A database of 44 writers with 48 samples of each digit totaling 21120 samples is used.

The ELM classifier is trained with 75% of the data (i.e. the first 33 writers) and tested by the remaining data (i.e. writers 34 to 44). A two-stage exhaustive parameter estimation technique is used to estimate the best values for SVM parameters (C and γ). The database is split into 4 subsets: three were used in training and validation in turn, and the fourth for testing. A large number of experiments were conducted to estimate the values for SVM parameters that produce the highest recognition rates. The estimated parameters were used in the training and testing of the SVM classifier.

In ELM, the number of hidden neurons that result in the highest recognition rates is estimated by using a training and verification technique. The highest recognition rates are achieved with 1000 hidden neurons. The recognition results of SVM and ELM are compared in terms of the recognition rates at the digit and writer level and in timing of training and testing. In addition, the results of the SVM and ELM classifiers are compared with the results of published work using Hidden Markov

Model (HMM) and the Nearest Mean (NM) classifiers. The achieved average recognition rates are 99.39%, 99.45%, 97.99% and 94.35% with SVM, ELM, HMM and NM classifiers, respectively. SVM and ELM recognition rates are better for all the digits. The classification errors are analyzed and categorized.

The presented technique, using simple features and SVM and ELM classifiers, proved to be effective in the recognition of Arabic numerals, and it was shown to be superior to HMM and NM classifiers for all digits.

The researchers are currently exploring the use of more statistical and syntactical features. The extension of the technique to Arabic text recognition and the use of multiple classifiers are under investigation.

Acknowledgment

We would like to thank the referees for their constructive criticism and stimulating remarks. The modification of the original manuscript to address those remarks improved the revised manuscript considerably. In addition, we would like to thank King Fahd University of Petroleum and Minerals for supporting this research work and providing the computing facilities.

8 References

[1] S. Mahmoud, “Recognition of Arabic (Indian) Numerals using Hidden Markov Models”, Signal Processing 88, April 2008, pp.844-857.

[2] C. Liu, K. Nakashima, H. Sako, H. Fujisawa, “Handwritten digit recognition: Benchmarking of state-of the- art techniques”, Pattern Recognition 36, (2003), pp. 2271–2285.

[3] M. Shi, Y. Fujisawa, T. Wakabayashi, F. Kimura, “Handwritten numeral recognition using gradient and curvature of gray scale image”, Pattern Recognition, 35, (2002), pp. 2051–2059.

[4] L.N. Teow, K.F. Loe,“Robust vision-based features and classification schemes for off-line handwritten digit recognition”, Pattern Recognition, 35, (2002), pp.2355–2364.

[5] K. Cheung, D. Yeung, R.T. Chin, “A Bayesian framework for deformable pattern recognition with application to handwritten character recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (12), (1998), pp.1382–1388.

[6] I.J.Tsang, I.R.Tsang, D.V. Dyck, “Handwritten character recognition based on moment features derived from image partition”, International Conference on Image Processing, vol. 2, (1998), pp. 939–942.

[7] F. Al-Omari,”Hand-written Indian numeral recognition systems using template matching approaches”, Proc ACS/IEEE Int. Conf Computer Syst. Appl., (2001);pp. 83–8.

[8] F. Al-Omari, O. Al-Jarrah, “Handwritten Indian numerals recognition system using probabilistic neural networks”, Advanced Engineering Informatics 18 (2004), pp. 9–16.

[9]F. Bousalma,”Structural and fuzzy techniques in the recognition of online Arabic characters”, Int. Journal Pattern Recognition and Artificial Intelligence, 13(7),(1999), pp.1027–1040.

[10] A. Salah, E. Albaydin, L. Akarun,”A selective attention-based method for visual pattern recognition with application to handwritten digit recognition and face recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002;24(3): pp. 420–425.

[11] A. Hamid, R. Haraty,”A neuro-heuristic approach for segmenting handwritten Arabic text”, Proc, ACS/IEEE Int. Conf Computer Syst. Appl., 2001; pp.110–113.

[12] S. Saloum,”Arabic hand-written text recognition”, Proc ACS/IEEE Int. Conf Computer Syst. Appl., 2001; pp.106–109.

[13] S. Almaadeed, C. Higgens, and D. Elliman, “Recognition of Off-line Handwritten Arabic Words using Hidden Markov Model Approach,” ICPR 2002, Quebec City, August 2002, pp. 481-484.

[14] S. Almaadeed, 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.

[15] S. Touj, N. Ben Amara,, H. Amiri, “Arabic Handwritten Words Recognition Based on a Planar Hidden Markov Model”, Int. Arab J. Inf. Technology, vol. 2, no. 4, pp. 318-325, 2005.

[16] B. Al-Badr, Sabri A. Mahmoud, “Survey and Bibliography of Arabic Optical Text Recognition,” J. of Signal Processing, Vol. 41, No.1, pp.49-77 (Jan. 1995).

[17] L. Lorigo, V. Govindaraju, “Offline Arabic Handwriting Recognition: A Survey”, EEE Trans. Pattern Analysis and Machine Intelligence, vol. 28, no. 5, pp. 712-724, May 2006.

[18] A. Nabawi and S. Mahmoud, "Arabic Optical Text Recognition: A Classified Bibliography" , Engineering Research Bulletin, Minufiyah University, Egypt, Vol. 23, No. 1, Jan. 2000, pp. 79-131.

[19] I. Bazzi, C. LaPre, J. Makhoul, and R. Schwartz, “Omnifont and Unlimited Vocabulary OCR for English and Arabic”, Proc. Int’l Conf. Document Analysis and Recognition, vol. 2, pp. 842-846, Ulm, Germany, 1997.

[20] I. Bazzi, R. Schwartz, and J. Makhoul,”An Omifont Open-Vacabulary OCR system for English and Arabic”, IEEE Transaction on PAMI, Vol. 21, No. 6, pp. 495-504, 1999.

[21] M. Shahrezea, K. F. Khotanzad, “Recognition of handwritten Persian/Arabic numerals by shadow coding and an edited probabilistic neural network”, Proceedings of the International Conference on Image Processing, vol. 3, (1995), pp. 436–439.

[22] H. Hosseini, A. Bouzerdoum, “A combined method for Persian and Arabic handwritten digit recognition”, Proceedings of the Australian New Zealand Conference on Intelligent Information Systems. (1996), pp. 80–83.

[23] F. Said, R. Yacoub, C. Suen,”Recognition of English and Arabic numerals using a dynamic number of hidden neurons”, Proceedings of the Fifth International Conference on Document Analysis and Recognition, (1999), pp. 237–240.

[24] J. Sadri, C.Y. Suen, T.D. Bui, “Application of support vector machines for recognition of handwritten Arabic/Persian digits”, Proceedings of Second Iranian Conference on Machine Vision and Image Processing, vol. 1, (2003), pp. 300–307.

[25] H. Soltanzadeh, M. Rahmati, “Recognition of Persian handwritten digits using image profiles of multiple orientations”, Pattern Recognition Letters 25 (2004) 1569–1576.

[26] V. Vapnik, *Estimation of Dependences Based on Empirical Data*, Addendum 1, New York: Springer-Verlag, 1982.

[27]V. Vapnik , *The Nature of Statistical Learning Theory,* Springer, N.Y. 1995.

[28] B. E. Boser, I. M. Guyon, and V. N. Vapnik,”A training algorithm for optimal margin classifiers”, In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144 -152 Pittsburgh, PA, 1992. ACM Press.

[29] C. Cortes and V. Vapnik,”Support vector networks”, M. Learning, 20:273-297, 1995.

[30] B. Scholkopf, C. Burges, and V. Vapnik,”Extracting support data for a given task”, In U. M. Fayyad and R. Uthurusamy, editors, Proceedings, First International Conference on Knowledge Discovery & Data Mining Menlo Park, 1995. AAAI Press.

[31] B. Scholkopf, C. Burges, and V. Vapnik,”Incorporating invariances in support vector learning machines”, In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendho, editors, Artificial Neural Networks –ICANN ’96, pp. 47 – 52, Berlin, 1996. Springer Lecture Notes in Computer in Computer Science, Vol. 1112.

[32] B. Sch¨olkopf, P. Simard, A. Smola, and V. Vapnik,"Prior knowledge in support vector kernels", In: Jordan M.I., Kearns M.J., and Solla S.A. (Eds.) Advances in Neural Information Processing Systems 10, MIT Press. Cambridge, MA, (1998), pp. 640–646.

[33] F. Camastra,"A SVM-based cursive character recognizer", Pattern Recognition 40, (2007), pp. 3721 – 3727.

[34] S. Mozaffari, K. Faez, and M. Ziaratban, "Structural decomposition and statistical description of Farsi/Arabic handwritten numeric characters," Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on, 2005, pp. 237-241 Vol. 1.

[35] H. Soltanzadeh and M. Rahmati, "Recognition of Persian handwritten digits using image profiles of multiple orientations," Pattern Recognition Letters, vol. 25, Oct. 2004, pp. 1569-1576.

[36] S. Mozaffari, K. Faez, and H. Kanan, "Feature comparison between fractal codes and wavelet transform in handwritten alphanumeric recognition using SVM classifier," Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, 2004, pp. 331-334 Vol.2.

[37] A. Mowlaei and K. Faez, "Recognition of isolated handwritten Persian/Arabic characters and numerals using support vector machines," Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on, 2003, pp. 547-554.

[38] V. Vapnik, *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

[39] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of International Joint Conference on Neural Networks (IJCNN2004), 25–29 July, 2004, Budapest, Hungary.

[40] G. Huang_, Q. Zhu, C. Siew, "Extreme learning machine: Theory and applications", Neurocomputing 70, 2006, pp. 489–501.

[41] G. Huang, Q. Zhu, K. Mao, C. Siew, P. Saratchandran, N. Sundararajan, "Can threshold networks be trained directly?" , IEEE Trans. Circuits Syst. II 53 (3) (2006b) 187–191.

[42] S. Mahmoud, http://faculty.kfupm.edu.sa/ICS/smasaad/Arabic%20Computing.htm

[43] HTK Speech Recognition Toolkit, http://htk.eng.cam.ac.uk/

[44] C. Hsu, C. Chang, "A Practical Guide to Support Vector Classification", www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.