To appear in Dirasat Engineering Sciences Journal, Vol. 37, No. 2, Oct 2010.

Feature Selection for Recognizing Handwritten Arabic Letters

Gheith A. Abandah and Tareq M. Malas

Computer Engineering Department, The University of Jordan, Amman 11942, Jordan

Tel: +962-6-5355000 ext. 23000

Fax: +962-6-5300813

E-mail: abandah@ju.edu.jo

Abstract There are many feature extraction methods for handwritten letters. These methods provide large sets of features that include redundant and irrelevant features. Feature selection is needed to select a subset of features that gives good recognition accuracy and has low computational overhead. We use feature selection techniques to evaluate a large set of features extracted from handwritten Arabic letters. We extract 96 features from the letter's secondary components, main body, skeleton, and boundary. These features are evaluated and best subsets of varying sizes are selected using five feature selection techniques. These techniques vary in complexity from selecting best individual features, through sequential forward selection, to evolutionary optimization algorithm. The best subsets of selected features include secondary components features, letter form, low-order elliptic Fourier descriptors, moments, size features, and features extracted from the boundary. We use three popular classifiers to evaluate the subsets selected by the five selection techniques and to find the recognition accuracy as a function of the feature subset size. The evolutionary algorithm has the highest time complexity but it selects feature subsets that give the highest recognition accuracies. In most cases, feature subset sizes of about 20 features achieve best recognition accuracy.

Keywords *Feature extraction, Feature evaluation, Feature selection, Pattern recognition, Handwritten Arabic letters*

1 Introduction

Recognition of handwritten cursive text such as Arabic text is an active research problem (Arica, 2002), (Lorigo, 2006). Offline recognition of unconstrained handwritten cursive text images must overcome many difficulties such as unlimited variation in human handwriting, similarities of distinct character shapes, character overlaps, and interconnections of neighboring characters. Although offline systems are less accurate than online systems that use touch screens or electronic pens, they are now good enough for specialized systems such as interpreting Latin handwritten postal addresses on envelopes and reading currency amounts on bank checks. Some progress has been made on recognizing handwritten Arabic text samples of limited vocabulary (e.g., IFN/ENIT database of handwritten Tunisian town names (Pechwitz, 2002)). In ICDAR Arabic handwriting recognition competitions held in 2005, 2007, and 2009 (Märgner, 2005), (Märgner, 2007), (Märgner, 2009), best systems' accuracies improved from 76% through 87% to 93% on the IFN/ENIT database. However, more progress is needed to achieve good accuracies on unlimited vocabulary.

Offline recognition of handwritten unconstrained cursive text involves several stages: preprocessing the scanned images to prepare them for latter stages, segmenting the scanned page into lines and words, segmenting the words into letters, extracting features from the segmented letters, recognizing these letters using a trained classifier, and post-processing to improve the recognition accuracy. Progress is needed in all these stages in order to build efficient recognition system.

In this paper, we concentrate on improving the feature extraction stage by selecting efficient feature subsets to extract. Figure 1 summarizes the methodology used in this paper. We use common feature extraction techniques (Lorigo, 2006) to extract a large set of features from a database of handwritten Arabic letter forms. We use five different feature selection techniques to select best feature subsets from the extracted 96 features. We evaluate these feature subsets to select a small subset of features that provides high recognition accuracy. We also analyze the recognition accuracy as a function of the feature subset size using three popular classifiers.

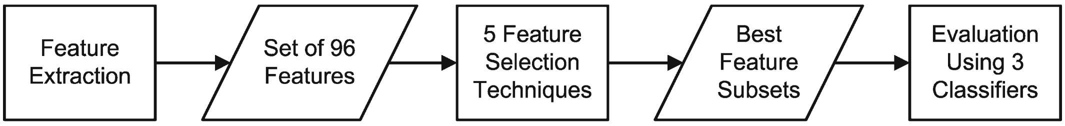

Feature Extraction
Set of 96 Features
5 Feature Selection Techniques
Best Feature Subsets
Evaluation Using 3 Classifiers

Fig. 1 Methodology of Feature Extraction, Selection, and Evaluation

Although there are some published work in selecting features for recognizing handwritten Arabic text (Pechwitz, 2006), (El Abed, 2007), (Abandah, 2008), (Abandah and Anssari, 2009), this work presents a comprehensive evaluation of large set of features using state-of-the-art feature extraction techniques.

This paper is organized in seven sections. Section 2 describes the five feature selection techniques used. Section 3 introduces the 96 features used in this research and describes their extraction techniques. Section 4 describes the three classifiers used to evaluate feature subsets. Section 5 describes the used database of Arabic letter samples and the feature extraction tools. It also provides the implementation details of the feature selection techniques. Section 6 describes the experiments and presents the results. This section evaluates the features and recommends best feature subsets. It also analyzes the classification accuracy as a function of the feature subset size. Finally, Section 7 provides a discussion and states the main conclusions, recommendations, and future work.

2 Feature selection

Feature selection is typically a search problem for finding an optimal or suboptimal subset of m features out of original M features. Feature selection is important in many pattern recognition problems for excluding irrelevant and redundant features. It allows reducing system complexity and processing time and often improves the recognition accuracy (Guyon, 2003). For large number of features, exhaustive search for best subset out of 2^M possible subsets is infeasible. Therefore, many feature subset selection algorithms have been proposed.

In this paper, feature subset selection is applied on a set of feature values x_{ijk} ; $i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, C$; and $k = 1, 2, \ldots, M$, where x_{ijk} is the *ith* sample of the *jth* letter form (class) of the *kth* feature. Therefore, the average of the *kth* feature for letter form ω_j is

$$\bar{x}_{jk} = \frac{1}{N}\sum_{i=1}^{N} x_{ijk} . \qquad (1)$$

And the overall average of the *kth* feature is

$$\bar{x}_{k} = \frac{1}{C}\sum_{j=1}^{C} \bar{x}_{jk} . \qquad (2)$$

The following subsections describe the feature subset selection techniques used in this paper.

2.1 Scatter criterion (*J*)

The simplest feature selection methods select *best individual features*. A feature evaluation function is used to rank individual features, then the highest ranked *m* features are selected. Although these methods can exclude irrelevant features, they often include redundant features. “The *m* best features are not the best *m* features” (Peng, 2005). One such feature evaluation function is the *scatter criterion* J_k , which is a ratio of the mixture scatter to the within-class scatter (Theodoridis, 2006). The *within-class scatter* of the *kth* feature is

$$S_{w,k} = \sum_{j=1}^{C} P(\omega_j)\, S_{jk} \text{ , } (3)$$

where S_{jk} is the variance of class ω_j , and $P(\omega_j)$ is the priori probability of this class and found by:

$$S_{jk} = \frac{1}{N}\sum_{i=1}^{N} (x_{ijk} - \bar{x}_{jk})^2 \text{ and } P(\omega_j) = \frac{1}{C} . \qquad (4)$$

The *between-class scatter* is the variance of the class centers with respect to the global center and is found by

$$S_{b,k} = \sum_{j=1}^{C} P(\omega_j)\, (\bar{x}_{jk} - \bar{x}_k)^2 \text{ . } (5)$$

And the *mixture scatter* is the sum of the within and between-class scatters, and equals the variance of all values with respect to the global center.

$$S_{m,k} = S_{w,k} + S_{b,k} = \frac{1}{CN}\sum_{j=1}^{C}\sum_{i=1}^{N} (x_{ijk} - \bar{x}_k)^2 \qquad (6)$$

Hence, the scatter criterion J_k of the *kth* feature is

$$J_k = \frac{S_{m,k}}{S_{w,k}} . \qquad (7)$$

Higher value of this ratio indicates that the feature has high ability in separating the various classes into distinct clusters.

2.2 Symmetric uncertainty (SU)

Another approach to select best individual features is to select the features that have highest *symmetric uncertainty* (SU) values between the feature and the target classes (Duda, 2001). To find this indicator, we first normalize the feature values for zero mean and unit variance by

$$\hat{x}_{ijk} = \frac{x_{ijk} - \bar{x}_k}{\sigma_k} \ , \ \sigma_k^2 = S_{m,k} = \frac{1}{CN}\sum_{j=1}^{C}\sum_{i=1}^{N}(x_{ijk} - \bar{x}_k)^2 \ . \qquad (8)$$

Then the normalized values of continuous features are discretized into L finite levels to facilitate finding probabilities. The corresponding discrete values are $\tilde{x}_{ijk}$. The *mutual information* of the *kth* feature is

$$I(\mathbf{x}_k, \boldsymbol{\omega}) = \sum_{l=1}^{L}\sum_{j=1}^{C} P(\tilde{x}_{ljk}, \omega_j)\log_2 \frac{P(\tilde{x}_{ljk}, \omega_j)}{P(\tilde{x}_{ljk})P(\omega_j)}, \qquad (9)$$

where $P(\tilde{x}_{lk})$ is the distribution of the *kth* feature and $P(\tilde{x}_{lk}, \omega_j)$ is the joint probability. This indicator measures how much the distribution of the feature values and target classes differ from statistical independence. This is a nonlinear estimation of correlation between the feature values and target classes. The *symmetric uncertainty* (SU) is derived from the mutual information by normalizing it to the entropies of the feature values and target classes.

$$SU(\mathbf{x}_k, \boldsymbol{\omega}) = 2\left(\frac{I(\mathbf{x}_k, \boldsymbol{\omega})}{H(\mathbf{x}_k) + H(\boldsymbol{\omega})}\right), \qquad (10)$$

where the entropy of variable X is found by $H(X) = -\sum_i P(x_i)\log_2 P(x_i)$.

2.3 Fast correlation-based filter (FCBF)

Many *sequential* and *random* search algorithms have been used in feature subset selection (Liu, 2005). The sequential search methods are variations of sequential forward selection, sequential backward elimination, and bidirectional selection. These algorithms are simple to implement and fast; they have time complexity of $O(M^2)$ or less. However, as they don't perform complete search, they may miss the optimal feature subset.

The *fast correlation-based filter* (FCBF) algorithm is a sequential forward selection algorithm and aims to select a subset of relevant features and exclude redundant features (Yu, 2004). FCBF uses the symmetric uncertainty $SU(\mathbf{x}_k, \boldsymbol{\omega})$ to estimate the relevance of feature k to the target classes. It also uses the symmetric uncertainty between two features k and o $SU(\mathbf{x}_k, \mathbf{x}_o)$ to approximate the redundancy between the two features. This algorithm grows a subset of *predominant* features by adding the relevant features to the empty set in descending $SU(\mathbf{x}_k, \boldsymbol{\omega})$ order. Whenever feature k is added, FCBF excludes from consideration for addition to the subset all remaining redundant features o that have $SU(\mathbf{x}_k, \mathbf{x}_o) \geq SU(\mathbf{x}_o, \boldsymbol{\omega})$. In other words, it excludes all features that their respective correlation with already selected features is larger than or equals their correlation with the target classes.

2.4 Minimal-redundancy-maximal-relevance (mRMR)

The *minimal-redundancy-maximal-relevance* (mRMR) algorithm is another sequential forward selection algorithm (Peng, 2005). It uses the mutual information to select best *m* features that have minimal redundancy and maximal relevance criterion.

For the complete set of features *X*, the subset *S* of *m* features that has the *maximal relevance* criterion is the subset that satisfies the maximal mean value of all mutual information values between individual features $\mathbf{x}_i$ and class $\boldsymbol{\omega}$.

$$\max D(S, \boldsymbol{\omega}), \quad D = \frac{1}{m} \sum_{\mathbf{x}_i \in S} I(\mathbf{x}_i, \boldsymbol{\omega}) \quad (11)$$

The subset *S* of *m* features that has the *minimal redundancy* criterion is the subset that satisfies the minimal mean value of all mutual information values between all pairs of features $\mathbf{x}_i$ and $\mathbf{x}_j$.

$$\min R(S), \quad R = \frac{1}{m^2} \sum_{\mathbf{x}_i, \mathbf{x}_j \in S} I(\mathbf{x}_i, \mathbf{x}_j) \quad (12)$$

In the mRMR algorithm, the subset *S* of *m* best features is grown iteratively using forward search algorithm. The following criterion is used to add the $\mathbf{x}_j$ feature to the previous subset of $m-1$ features:

$$\max_{\mathbf{x}_j \in X - S_{m-1}} \left[I(\mathbf{x}_j, \boldsymbol{\omega}) - \frac{1}{m-1} \sum_{\mathbf{x}_i \in S_{m-1}} I(\mathbf{x}_i, \mathbf{x}_j) \right] \quad (13)$$

2.5 Non-dominated sorting genetic algorithm (NSGA)

Genetic algorithms are random search algorithms and often offer efficient solutions to general NP-complete problems. They can explore large, nonlinear search space by performing simultaneous search in many regions. A population of solutions is evaluated using some fitness function. In feature selection, this fitness function usually calls the classifier to evaluate the population's individuals (feature subsets); constituting a wrapper algorithm. The individuals' fitness is then used to select individuals for breeding and producing the next generation. *Multi-objective genetic algorithms* (MOGA) have been successfully used in feature selection (Oliveira, 2003). MOGA have the advantage of generating a set of alternative solutions.

The *non-dominated sorting genetic algorithm* (NSGA) is an efficient algorithm for multi-objective evolutionary optimization (Srinivas, 1995). We use NSGA to search for optimal set of solutions with two objectives:

i. Minimize the number of features used in classification.
ii. Minimize the classification error.

This algorithm searches for a set of optimal solutions on a front called the *Pareto-optimal front*. Figure 2 shows an example Pareto-optimal front and a population of solutions found in optimizing the number of features and the classification error. This front is the set of *non-dominated* solutions among this population. A non-dominated solution is one that does not have any other solution that dominates it. Solution $S^{(1)}$ dominates Solution $S^{(2)}$ when no objective value of $S^{(2)}$ is less than $S^{(1)}$ and at least one value of $S^{(2)}$ is strictly greater than $S^{(1)}$. In this two-objective case, a non-dominated solution of *m* features is the solution that has the smallest classification error among all solutions that have *m* features.

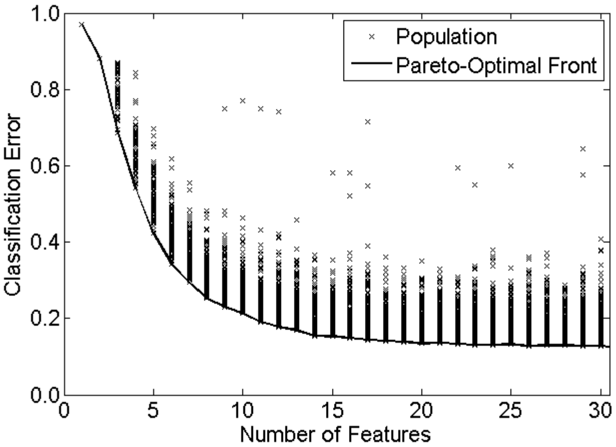


Fig. 2 Example Pareto-Optimal Front and Population Examined by NSGA

3 Feature set

There are many used feature extraction methods for offline recognition of characters. These methods are extracted from the character's binary image, boundary, or skeleton (Trier, 1996), (Dalal, 2008). The following subsections describe an assortment of features used in this research. The letter form feature is described after giving a necessary overview of the Arabic writing. Then we describe our approach that starts by detecting the secondary components of the Arabic letters and extracting features from these components. Then the secondary components are removed and additional features are extracted from the main body, the main body's skeleton, and the main body's boundary.

3.1 Overview of Arabic writing

Arabic is written from right to left and is always cursive. The Arabic alphabet has 28 basic letters (Abandah and Khedher, 2009). Each letter has multiple forms depending on its position in the word. Each letter is drawn in an isolated form when it is written alone, and is drawn in up to three other forms when it is written connected to other letters in the word. For example, the letter **Ain** has four forms: *isolated* (ع), *initial* (عـ), *medial* (ـعـ), and *final* (ـع). Within a word, every letter can connect from the right with the previous letter. However, there are six letters that do not connect from the left with the next letter. These letters have only the isolated and final forms. When one of these six letters is present in a word, the word is broken into *sub-words*, often called *parts of Arabic word* (PAWs). For example, the word "Arabic" (عربية) has two PAWs: the first PAW consists of initial **Ain** (عـ) and final, leftdisconnecting, **Reh** (ر); and the second PAW consists of initial **Beh** (ﺑ), medial **Yeh** (ﻴ), and final **Teh Marbuta** (ﺔ).

3.2 Letter form

When segmenting PAWs into letters, the letter form feature can be easily found. The letter form of a letter in a single-letter PAW is isolated. The letter forms of the first letter and last letter in a multi-letter PAW are initial and final, respectively. Finally, the letter form of a non-boundary letter in a PAW with more than two letters is medial.

3.3 Secondary components detection and removal

More than half the Arabic letters are composed of *main body* and *secondary components*. The secondary components are letter components that are disconnected from the main body. For example, **Beh** (ب) has a dot under its main body, **Teh** (ت) has two dots above its main body, and **Kaf** (ك) has a zigzag enclosed within the main body.

Detecting the secondary components can be done after segmenting the binary image of the letter into its disconnected components using the *connected component labeling* techniques (Rosenfeld, 1976). Then the main body is easily identified as it is usually the largest component and is closer to the letter's center than the secondary components. The *secondary position* is then easily found as the position of the secondary components relative to the main body. Finally, the number and position of the secondary components play important role in finding the *secondary type*. However, our approach in finding the type of the secondary components also utilizes other features extracted from the secondary components such as size, orientation, roundness, and spatial distribution (see Section 3.4).

After detecting and classifying the secondary components, we remove them from the letter image and pass the main body to the other feature extraction stages described below.

3.4 Main body features

Main body features are mainly statistical features. They are found from the letter image after removing the secondary components. The following paragraphs define some of these features.

Size. We use a threshold function to convert the 2-dimentional image into a binary image $B(x,y)\in[0,1]$; black pixels are the foreground pixels and take the value 1 (Jain, 1995). A low threshold is used to maintain connectivity of light pen strokes. The *area A* of the letter body is found by

$$A=\sum_{x}\sum_{y}B(x,y)\,. \qquad (14)$$

To find the main body's *width W* and *height H*, the image is clipped into a rectangular shape such that all four borders have at least one black pixel. We also derive a scale-invariant feature; the *width to height ratio W/H* (Trier, 1996).

Distribution. We partition the clipped image into four equal quadrants and find the fraction of black pixels in each quadrant relative to the area *A*. The resulting four fractions are: upper-right *UR/A*, lower-right *LR/A*, lower-left *LL/A*, and upper-left *UL/A*. We also find the fractions of the four halves relative to *A*: upper *U/A*, right *R/A*, lower *Lo/A*, and left *Lt/A*.

Moments. The *normalized central moments* of order $(u+v)$ of the binary image, which are translation and scale invariant (Theodoridis, 2006), (Reiss, 1991), are found by

$$\eta_{uv} = \frac{1}{A^k} \sum_x \sum_y (x-\bar{x})^u (y-\bar{y})^v B(x,y), \qquad (15)$$

where $k = 1 + (u+v)/2$ for $u+v \geq 2$.

We compute the normalized center of mass $(\bar{x}_N, \bar{y}_N)$ from the image's center of mass $(\bar{x}, \bar{y})$ using

$$\bar{x}_N = \frac{\bar{x} - (W-1)/2}{W/2} \text{ and } \bar{y}_N = \frac{\bar{y} - (H-1)/2}{H/2}. \qquad (16)$$

Orientation. The *orientation* θ of an elongated object is the orientation of the elongation axis (Jain, 1995). The axis of least inertia is the elongation axis. The inertia of the elongation axis is found by

$$\chi^2 = \sum_x \sum_y r^2 B(x,y), \qquad (17)$$

The orientation θ then is found by solving

$$\sin 2\theta = \pm \frac{2\mu_{11}}{\sqrt{4\mu_{11}^2 + (\mu_{20} - \mu_{02})^2}} \text{ and} \qquad (18)$$

$$\cos 2\theta = \pm \frac{(\mu_{20} - \mu_{02})}{\sqrt{4\mu_{11}^2 + (\mu_{20} - \mu_{02})^2}}, \qquad (19)$$

where μ_{uv} is central moment of order $(u+v)$.

Roundness. The positive and negative values for the sine and cosine of 2θ in Eqs. (18) and (19) are used to find the minimum and maximum inertia values, respectively. The object *elongation E* (or eccentricity) is $E = \chi_{\max} / \chi_{\min}$. The object *roundness* $R = \chi_{\min}^2 / \chi_{\max}^2$ is a ratio between 0 for a straight line and 1 for a circle.

Loops. The number of main body loops is a structural feature. There are many techniques to find the number of loops in an image. We use the connected component labeling algorithm to find the number of loops. The number of background components (white components) minus one is the number of loops. For example, **Sad** (ص) has one loop because it has two background components: the large background component surrounding the letter (always present) and the small component enclosed within the loop in the right.

3.5 Skeleton features

Thinning is usually a pre-processing stage in character recognition where the character image is reduced to a simplified one-pixel wide skeleton. We use Deutsch's thinning algorithm which gives good skeletons for our samples (Deutsch, 1972). We use the skeleton of the main letter’s body to extract the following five features.

Vertical and horizontal crossings. The *vertical* and *horizontal crossings* are found by counting the number of white-black-white transfers when scanning the image’s pixels on a vertical line and a horizontal line, respectively. These lines are the two lines that pass through the center of mass of the main body’s skeleton.

Feature points. Three important feature points can be easily found from the skeleton by examining the eight immediate neighbors of every black pixel: *end point* is a point with one black neighbor, *branch point* has three black neighbors, and *cross point* has four black neighbors.

3.6 Boundary features

Boundary finding is another pre-processing stage in character recognition where the character outer contour is found (Ha, 1997). We find the boundary of the main letter’s body and use it to extract the following six features.

Boundary pixels. The number of *boundary pixels m* is directly found by counting the boundary pixels $(x_i, y_i), i = 1, 2, \ldots, m$. Then Freeman chain code is used to compactly encode the boundary pixels (Freeman, 1961). The direction from every boundary pixel to the next boundary pixel is put in the chain. The direction from the last pixel to the first pixel is the last code in the chain. The direction codes $f_i \in [0, 7]$ are used such that right is 0, up-right is 1, up is 2, *etc*.

Perimeter length. The *perimeter length T* is found by summing the distances from one pixel to the next. Formally, it is found from the chain code using

$$T = \sum_{i=1}^{m} \mathrm{L}(f_i), \quad \text{where} \quad \mathrm{L}(f_i) = \begin{cases} 1 & f_\mathrm{i} \text{ is even} \\ \sqrt{2} & f_\mathrm{i} \text{ is odd} \end{cases}. \qquad (20)$$

Perimeter to diagonal ratio. We also use a scale-invariant feature which is the ratio of half the perimeter length to the diagonal of the clipped main body rectangle *T/2D*. For simple shapes like **Alef** (ا), this ratio is 1, and this ratio is larger than 1 for more complex shapes.

$$T/2D = \frac{T/2}{\sqrt{W^2 + H^2}} \qquad (21)$$

Compactness ratio. Another derived feature from the perimeter length and the area is the *compactness ratio* or roundness ratio which is found by Eq. (22) (Theodoridis, 2006).

$$\gamma = \frac{T^2}{4\pi A} \qquad (22)$$

This ratio is 1 for a filled circle and is larger than 1 for distributed complex shapes.

Bending energy. The *bending energy E* is a measure of the curvature of the boundary (Theodoridis, 2006). It can be found from the chain code by summing the squares of the direction changes from one boundary pixel to the next.

$$E = \frac{1}{T}\sum_{i=1}^{m}\left(\frac{\pi}{4}\times \mathrm{IF}(k_i > 4, 8-k_i, k_i)\right)^2 , \qquad (23)$$

$$\text{where} \quad k_i = \begin{cases} \mathrm{mod}(f_{i+1} - f_i, 8) & i < m \\ \mathrm{mod}(f_1 - f_m, 8) & i = m \end{cases}. \qquad (24)$$

Elliptic Fourier descriptors. The piecewise linear curve that passes through all boundary pixels is a closed outer contour curve. This curve can be approximated using the *elliptic Fourier descriptors* (EFD) (Kuhl, 1982). These descriptors are useful features (Trier, 1996), (Mezghani, 2002) and are used to approximate the curve.

4 Classifiers

To ensure that our results are not restricted to a specific classifier, we use three widely-used classifiers: *k*-nearest neighbor (*k*-NN), linear discriminant analysis (LDA), and support vector machine (SVM) (Duda, 2001). These classifiers are often used in evaluating various feature sets (Peng, 2005), (Wei, 2007). Therefore, we expect that the selected features give good accuracy on various types of classifiers. These classifiers are usually trained using *n* training samples. Each training sample $\mathbf{x}_i; i = 1, 2, \ldots, n$, is a vector of *m* feature values of a known class. Given a testing sample $\mathbf{x}_j$ of an unknown class, the classifier finds the class of this sample. These three classifiers are described below.

***k*-Nearest Neighbor (*k*-NN):** This classical classifier classifies $\mathbf{x}_j$ by assigning it the class most frequently represented among the *k* nearest training samples (Mitchell, 1997). Neighborhood is found based on a distance metric. We found that best results are achieved with $k = 5$ and using the city block distance metric. The data is first scaled to zero mean and one standard deviation.

Linear Discriminant Analysis (LDA): The LDA classifier is one of the earliest classifiers (Webb, 2002). It learns a linear classification boundary for the training samples space. It can be used for both 2-class and multiclass problems. LDA fits a multivariate normal density to each class, with a pooled estimate of covariance. In order to overcome the problems of high dimensionality and co-linearity, we use the *principal component analysis* as a preprocessing stage (Fukunaga, 1990).

Support Vector Machine (SVM): SVM is a newer classifier that uses kernels to construct linear classification boundaries in higher dimensional spaces (Burges, 1998). SVM selects a small number of critical boundary samples from each class and builds a linear discriminant function. The SVM used in this paper is the LIBSVM package (Hsu, 2002). Using grid search, we found that best results are achieved with the RBF kernel (radial basis function), penalty parameter $C = 12$, and gamma parameter $\gamma = 0.04$. Similar to *k*-NN, the data is first scaled to zero mean and one standard deviation.

5 Implementation

Our experimental setup comprises a database of handwritten Arabic samples and feature extraction, selection, and evaluation tools. The following subsections describe this database and present the implementation details of the feature selection tools.

5.1 Database of handwritten Arabic samples

Our database of handwritten Arabic samples was collected from 48 volunteers (Khedher, 2002). These volunteers were selected to represent various age, gender, and educational background groups. The samples were collected by asking the participants to write, as they normally do write, on a blank paper a one page of cursive Arabic text. This text was carefully selected so that it contains all the letter forms of the 28 Arabic letters.
We have extracted from the 48 page samples about 440 collections of individual words, PAWs, and letter forms. Each collection comprises 48 samples from 48 different volunteers. Figure 3 shows the collection of 48 samples of the isolated **Ain** form.

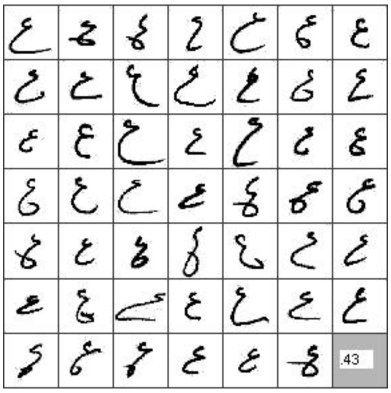
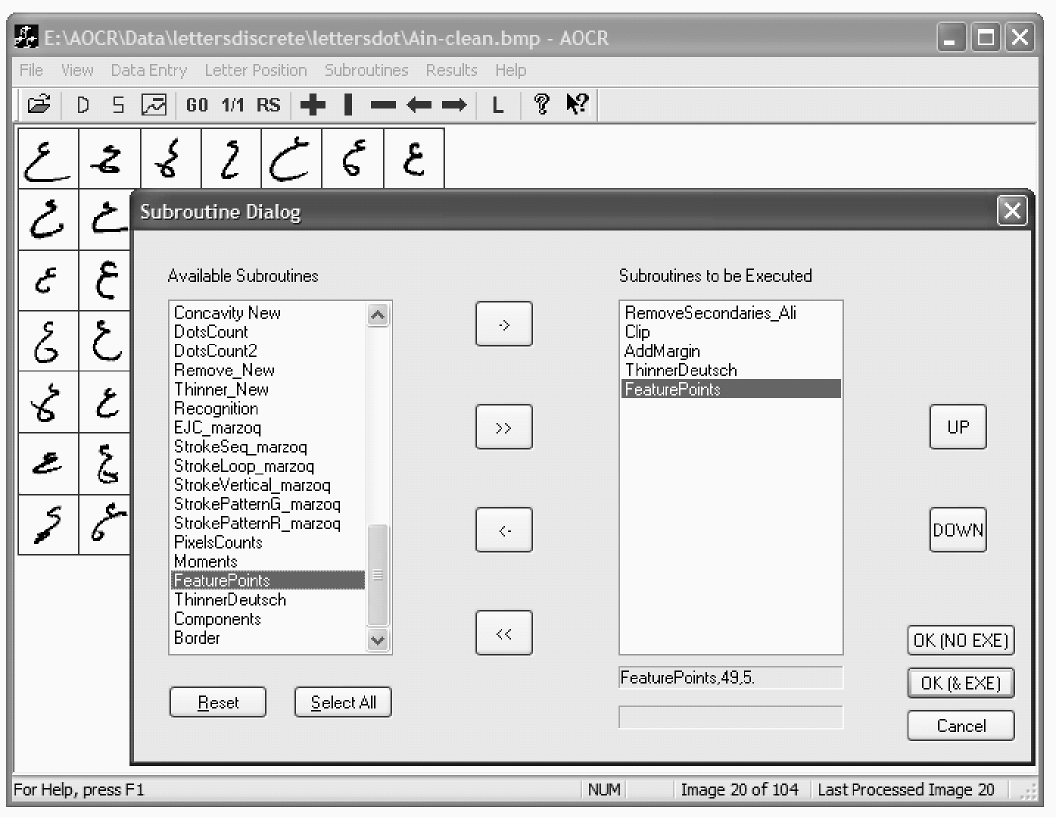


Fig. 3 A Collection of 48 Samples of the Isolated **Ain** Form

The collections of the initial, medial, and final letter forms were extracted after manually segmenting their cursive PAWs into individual letters. Manual segmentation is used to avoid errors that may come from an automatic letter segmentation process. We use in this research 104 collections of letter forms: 30 isolated forms, 22 initial forms, 22 medial forms, and 30 final forms. These collections contain all the 28 basic Arabic letters.

5.2 Feature extraction tools

To allow easy extraction of many features from this database of handwritten Arabic samples, we developed a desktop application. This application is an expandable tool that allows developers to easily add various preprocessing and feature extraction routines. It enables the user to select the order of the routines to be applied on the sample collections. This application allows the user to visualize the results of preprocessing routines and obtain the results of the feature extraction routines. Figure 4 shows this application with its dialog box for selecting what routines to apply on the collection of the isolated **Ain** samples.



E:\AOCR\Data\lettersdiscrete\lettersdot\Ain-clean.bmp - AOCR

Subroutine Dialog

Available Subroutines

Concavity New
DotsCount
DotsCount2
Remove_New
Thinner_New
Recognition
EJC_marzoq
StrokeSeq_marzoq
StrokeLoop_marzoq
StrokeVertical_marzoq
StrokePatternG_marzoq
StrokePatternR_marzoq
PixelsCounts
Moments
FeaturePoints
ThinnerDeutsch
Components
Border

Subroutines to be Executed

RemoveSecondaries_Ali
Clip
AddMargin
ThinnerDeutsch
FeaturePoints

FeaturePoints,49,5.

For Help, press F1 | NUM | Image 20 of 104 | Last Processed Image 20

Fig. 4 Arabic OCR Application Tool and Its Routine Selection Dialog Box

The preprocessing routines implemented in this application include binarization, noise removal, thinning, and boundary finding. This application also features batch processing where the selected routines can be applied on multiple sample collections. The results of the feature extraction routines can be exported from this application into an Excel spreadsheet. We have implemented in this application feature extraction routines for all the features described in Section 3. These routines were applied on the 104 collections of letter forms.

5.3 Feature selection tools

Using the feature extraction application described in the previous subsection, we extracted 96 features for the $104 \times 48 = 4{,}992$ letter samples. This data is organized in a spreadsheet and is fed to a tool written in C++ for feature processing. This tool performs needed normalization, discretization, and probability estimation, as described in Section 2, to find the scatter criterion and symmetric uncertainty. This tool also selects features using the FCBF algorithm and reports them in descending SU values order.

We used the C/C++ implementation of the mRMR algorithm developed by Peng *et al*. (Peng, 2005). The data is fed to the mRMR program after normalization and discretization.

We used a fast implementation of the multi-objective genetic algorithms (NSGA-II) developed by Illinois Genetic Algorithms Laboratory (Deb, 2002). By performing grid search, we selected the following parameter settings:

- Population size: 128
- Number of generations: 1000
- Selection type: tournament of size 2 without replacement
- Crossover probability (p_c): 0.8 using simulated binary crossover and 0.8 gene-wise swap probability
- Probability of mutation (p_m): 0.1, selective

NSGA is used to find solutions that optimize the dual objectives: classification accuracy *A* and feature subset size *m*. To evaluate the fitness of an individual, the NSGA program calls one of the three classifiers. Given a subset of *m* features, the classifier returns the classification accuracy. We prepared three MATLAB programs for the three classifiers that we refer to as NSGA/*k*-NN, NSGA/LDA, and NSGA/SVM.

The relation between best *A* and *m* is not monotonic. The accuracy increases as *m* increases for small *m*, and it often falls after some high *m* value. Therefore, the NSGA program does not explore the search space evenly. It concentrates on the search subspace of small *m* values. To overcome this problem, we modified the value returned by the classifier to include a dummy term proportional to *m*. The retuned dummy fitness ($A + 0.02m$) is monotonic and ensures that the NSGA program searches evenly for all *m* values.

To reduce execution time, we only use half of the available samples in the NSGA experiments. The 4-fold cross validation method is used to avoid over-fitting and to get stable results (Stone, 1974).

6 Experiments and results

This section presents the results of the conducted experiments. We start by presenting the results of evaluating the 96 features using two metrics. Then we present the results of the five feature selection techniques and present the best subset of 20 features. Finally, we present the results of evaluating best subsets of *m* features using the three classifiers.

6.1 Feature Evaluation

Figure 5 shows the scatter criterion for the 96 features found from all the sample data. Higher values indicate that the respective features have high ability in clustering the samples of distinct letters into distinct clusters.

Figure 6 shows the symmetric uncertainty for the 96 features. Higher values indicate that the respective features have high correlation with the target classes. Similar to the scatter criterion results, the symmetric uncertainty of the letter form, secondaries features, size-related features, and some EFDs is high. The features extracted from the skeleton and the bending energy have relatively low symmetric uncertainty. Moreover, orientation and features extracted from the boundary have relatively high symmetric uncertainty values.

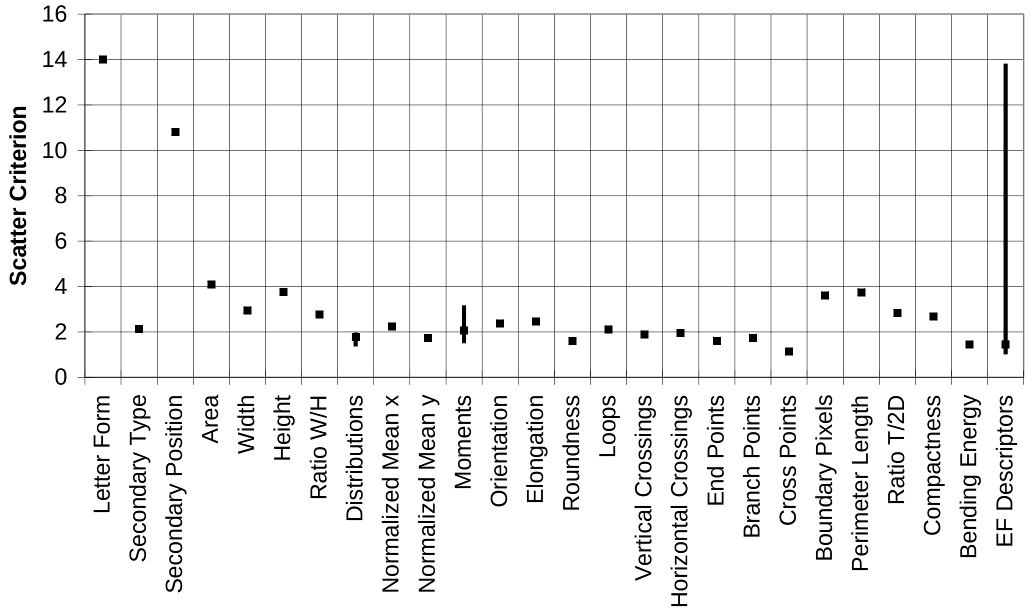


Fig. 5 Scatter Criterion for 96 Features. Features distributions, moments, and EF descriptors are sets of features that have averages and ranges.

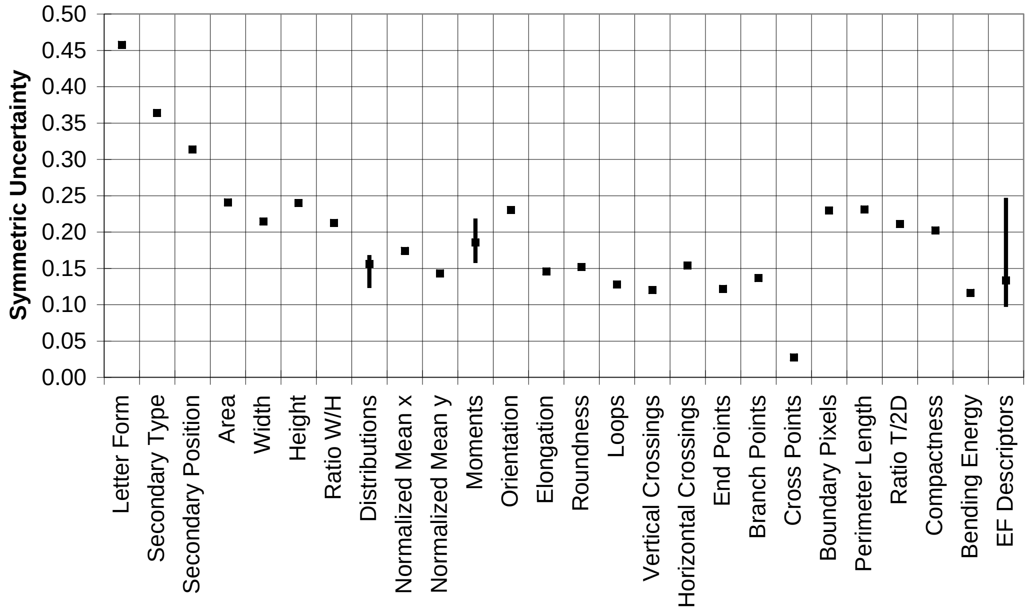


Fig. 6 Symmetric Uncertainty for 96 Features

6.2 Best 20 features

Using the five feature selection techniques, we found the best subset of 20 features. These five 20-feature subsets are shown in Table 1 and include the letter form, secondary type and position features, several low-order EFDs, few normalized central moments, and some statistical features extracted from the main body or the boundary, e.g., area, orientation, and perimeter to diagonal ratio. Only one skeleton feature (vertical crossing) makes it to this list, in the NSGA/LDA's subset.

There are many common features in the five subsets; especially the subsets of scatter criterion, symmetric uncertainty, and mRMR. In fact 17 out of the 20 features are common in these three subsets. The feature subset selected by the FCBF algorithm has only 14 features common with the mRMR's subset. This low number is a result of excluding some important features such as secondary position and perimeter length. These features are excluded because they have relatively high correlation with selected features. Although the features found by NSGA are found through optimizing the classification error for the LDA classifier, they include 16 features that are common with the subsets of the other four feature selection methods.

The rightmost two columns in Table 1 summarize the results of these five feature selection methods and the results found when using NSGA with the *k*-NN and SVM classifiers (not shown in this table) in addition to the LDA classifier. These two columns list the 20 most common features using the seven feature selection methods and the frequency of how many times the respective features are selected using the seven methods. The features secondary type, letter form, and EFDs a_0, c_0, and c_1 are unanimously selected by the seven methods.

Table 1 Best Twenty-Feature Subsets Found Using Various Methods

Order	Scatter Criterion	Symm. Uncertainty	FCBF	mRMR	NSGA/LDA	**Best Features**	Frequency
1	Letter Form	Letter Form	Letter Form	Secondary Type	Secondary Type	**Secondary Type**	7
2	Descriptor a_0	Secondary Type	Secondary Type	Letter Form	Letter Form	**Letter Form**	7
3	Secondary Position	Secondary Position	Descriptor a_0	Area	Secondary Position	**Descriptor a_0**	7
4	Area	Descriptor a_0	Area	Orientation	Descriptor a_0	**Descriptor c_0**	7
5	Height	Area	Height	Secondary Position	Descriptor c_0	**Descriptor c_1**	7
6	Perimeter Length	Height	Descriptor a_1	Descriptor a_0	Descriptor c_1	**Area**	6
7	Boundary Pixels	Descriptor a_1	Descriptor c_1	Descriptor a_1	Height	**Secondary Position**	6
8	Descriptor c_1	Perimeter Length	Orientation	Descriptor c_0	Perimeter Length	**Height**	6
9	Moment η_{02}	Descriptor c_1	Width	Descriptor c_1	Descriptor b_1	**Descriptor b_1**	6
10	Descriptor c_0	Orientation	Descriptor c_0	Height	Width	**Compactness**	6
11	Width	Boundary Pixels	Descriptor b_1	Perimeter Length	Descriptor c_2	**Width**	5
12	Ratio *T/2D*	Moment η_{02}	Ratio *T/2D*	Descriptor b_1	Normalized Mean *x*	**Normalized Mean *x***	5
13	Ratio *W/H*	Width	Moment η_{03}	Width	Boundary Pixels	**Orientation**	4
14	Compactness	Descriptor c_0	Compactness	Ratio *T/2D*	Compactness	**Perimeter Length**	4
15	Moment η_{20}	Ratio *W/H*	Moment η_{20}	Moment η_{02}	Horizontal Cross.	**Ratio *T/2D***	4
16	Elongation	Descriptor b_1	Descriptor c_2	Descriptor c_2	Descriptor a_2	**Moment η_{02}**	4
17	Orientation	Ratio *T/2D*	Descriptor b_2	Normalized Mean *x*	Moment η_{12}	**Descriptor c_2**	4
18	Normalized Mean *x*	Moment η_{03}	Descriptor d_1	Boundary Pixels	Fraction *U/A*	**Boundary Pixels**	4
19	Secondary Type	Compactness	Moment η_{03}	Ratio *W/H*	Loops	**Ratio *W/H***	4
20	Loops	Moment η_{20}	Descriptor a_2	Moment η_{20}	Descriptor d_3	**Moment η_{20}**	4

6.3 Classification accuracy

To find how many features are needed to achieve good character recognition accuracy, we find the classification error as a function of the number of features used. In each experiment, we used best *m* features as selected by the five feature selection methods; for *m* = 4, 5, …, 96. The results are shown in Figure 7. For every feature selection method, we evaluated the best *m* features using the 10-fold cross validation method on the *k*-NN, LDA, and SVM classifiers. The feature subsets used in the three NSGA curves come from respective optimizing experiments with NSGA/*k*-NN, NSGA/LDA, and NSGA/SVM. Note that the curves of the FCBF method stop at $m = 79$ because this method excludes features as discussed earlier.

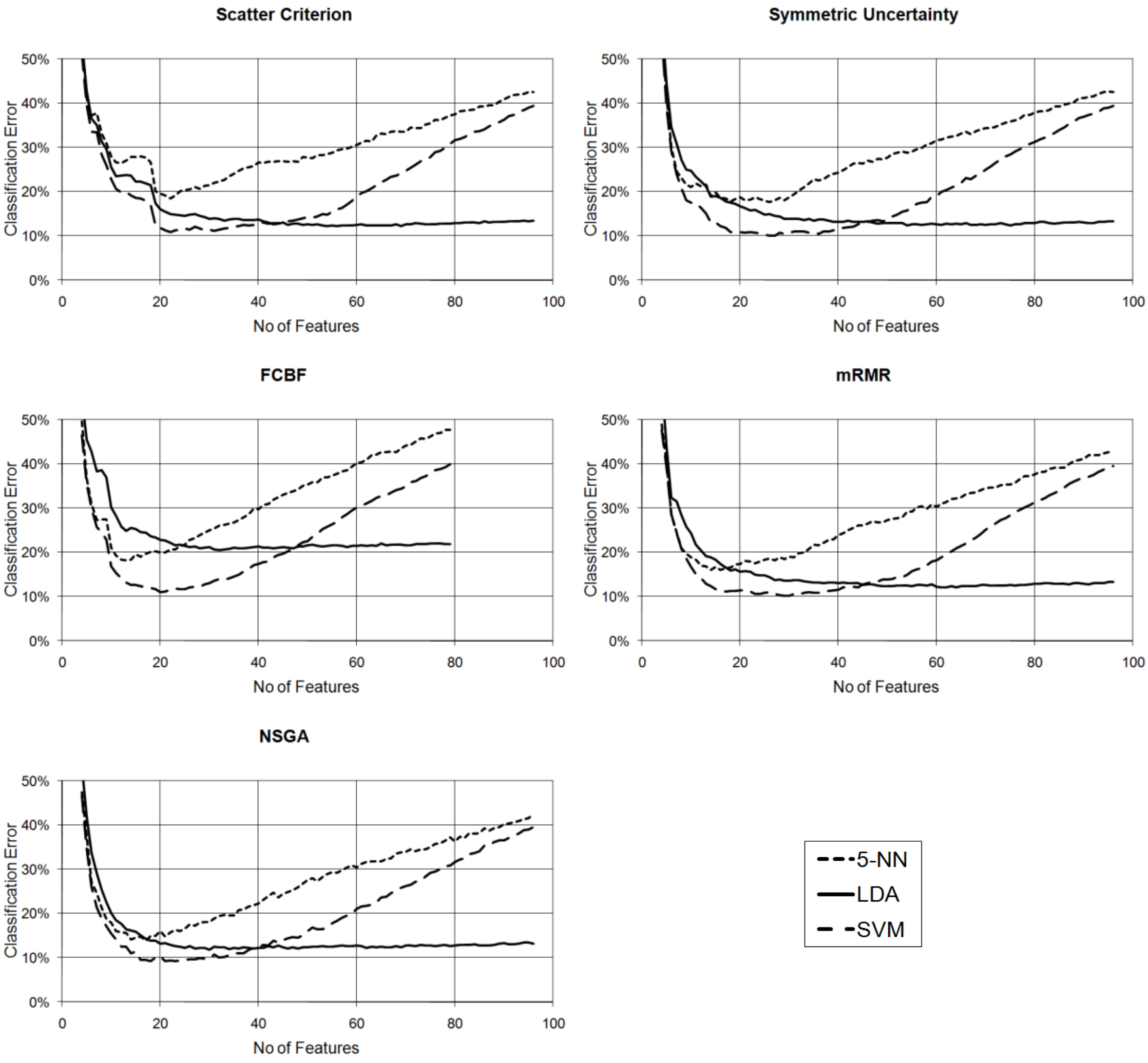


Fig. 7 Classification Error of the Feature Subsets Selected by the Five Feature Selection Methods on the Three Classifiers

For the three classifiers, the classification error decreases fast as the number of features increases from 4 to about 20 features. The LDA's classification error keeps decreasing slowly with more features. However, the *k*-NN's classification error increases when the number of features increases after reaching a minimum value in the region $m \in [13, 26]$ depending on the feature selection method used. SVM's classification error also increases with large *m* values, but stays with low values in a larger *m* region. For small *m* values, best classification accuracy is achieved by the SVM classifier. However, as SVM's classification error increases with large *m* values, the best accuracy is achieved by the LDA classifier for large *m* values. The SVM classifier achieves best classification accuracy for 20 features. And best SVM classifier's results are achieved using features selected by the NSGA/SVM method. Figure 8 gives clearer comparisons among the five feature selection methods on the three classifiers. This figure shows best results achieved for every feature selection method/classifier combination for $m \leq 20$ features. The NSGA/SVM and SVM combination achieves the lowest classification error of 9% at $m = 18$ features.

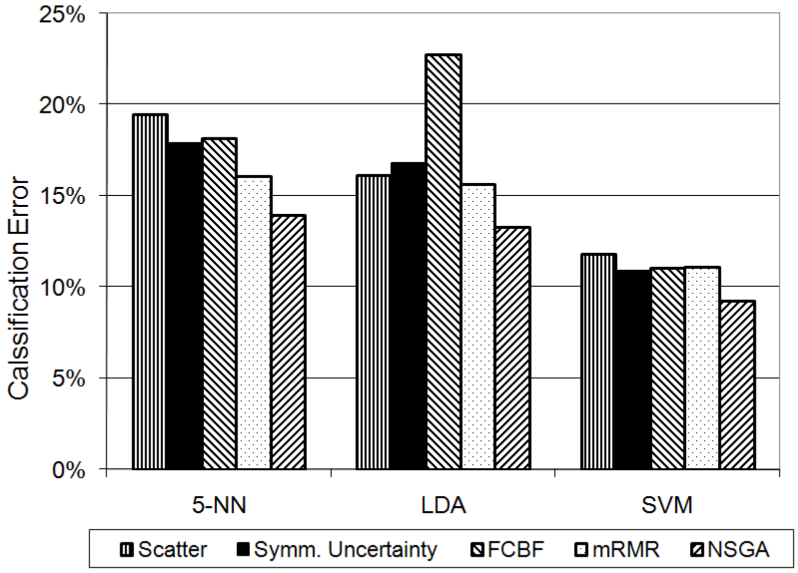


Fig. 8 Classification Error Using 20 Best Features Selected by the Five Selection Methods on the Three Classifiers

In general, best results are achieved with the features selected by the NSGA method followed by mRMR method. The FCBF and scatter criterion methods give unreliable results compared with the other three methods. The FCBF method selects features that give the worst classification error (23% with the LDA classifier). The scatter criterion method selects features that give the worst classification error when using the *k*-NN and SVM classifiers. Also note that the SVM classifier has best classification accuracy and the *k*-NN classifier has the worst.

7 Discussion and Conclusions

In this discussion, we try to tackle the following questions:

- Is the set of 96 features used big enough? Do we miss any important feature that captures some letter features not captured by these 96 features? Can the results be improved by adding other features?
- Are features selected by an NSGA method that uses certain classifier for fitness evaluation useful when using a different classifier in the production recognition system?

We used the feature subset that gave best results in Figure 8 with an SVM classifier and examined the recognition accuracy of every letter form. We found that the classification error is between 0% for easy to recognize letter forms and 29% for the hardest letter forms. The error is 0% for easy to recognize letters such as **Alef** (ا), **Yeh** (ي), and **Thal** (ذ). Table 2 shows the ten letter forms that have the worst classification errors.

Table 2 Letter of the Worst Ten Classification Errors

No	Letter	Error	Often Mistaken For
1	Isolated **Feh** (ف)	29%	ق ض
2	Medial **Hah** (ـحـ)	29%	ـعـ ـمـ
3	Medial **Feh** (ـفـ)	29%	ـقـ ـغـ
4	Medial **Meem** (ـمـ)	29%	ـعـ ـحـ
5	Medial **Ghain** (ـغـ)	27%	ـفـ
6	Isolated **Noon** (ن)	25%	ف
7	Medial **Ain** (ـعـ)	25%	ـصـ ـمـ
8	Final **Hah** (ـح)	21%	ع
9	Final **Qaf** (ـق)	21%	ص
10	Isolated **Zah** (ظ)	19%	ط

These ten letters are always drawn with loops or drawn with loops in some writing variations (Abandah and Khedher, 2009). There are substantial similarities among multiple Arabic letter form groups that have loops. Often the sole difference between such letters is a subtle difference in the loop's shape; notice how the medial **Feh** (ـفـ) and medial **Ghain** (ـغـ) differ. Moreover, letters with dots above the main body tend to have low recognition accuracies because the variations in drawing the dots give inaccuracies in extracting the secondary type feature (Abandah and Khedher, 2009). After careful examination of the samples that were incorrectly recognized, we concluded that most of these samples are hard to recognize even by a human expert reader. However, we think that the door is open to search for extracting new features that capture subtle differences in loop shapes and secondary types.

Referring to Figure 8, the average classification error of the three feature subsets selected by NSGA/*k*-NN, NSGA/LDA, and NSGA/SVM on the classifiers *k*-NN, LDA, and SVM, respectively, is 12.1%. We carried additional experiments where the feature subset selected by one NSGA/classifier program is used on the other two classifiers and the classification error is recorded. For these six cases, the average classification error is 13.0%. The average classification error of the second best selection method (mRMR) is 14.3%. These numbers suggest that the features selected by an NSGA method that uses certain classifier are useful when using a different classifier in the recognition system.

In this paper, we have presented the results of using five feature selection methods to select feature subsets to recognize handwritten Arabic letters. These methods select out of 96 extracted features some important features such as the secondary type and position, letter form, and low-order elliptic Fourier descriptors. Other often selected features are moments and size-related features. Subset sizes of 20 features or less are generally sufficient to achieve low classification error.

The genetic algorithm method that wraps a classifier selects best subsets of features compared with the other four methods. The next best method is mRMR. For the purpose of selecting features for recognizing handwritten Arabic letters, we recommend using the genetic algorithm method. However, if one cannot afford its complexity and long execution time, mRMR provides a good alternative.

Our future work includes using the best feature subset indentified in this research in a complete system for recognizing handwritten Arabic text. Our approach segments the sub-words into graphemes prior to feature extraction and classification. Additionally, we plan to evaluate some features that were successfully used with oriental languages such as chain code and gradient features (Liu, 2008).

Acknowledgements

This work was supported in part by the Deanship of Academic Research, The University of Jordan.

References

Abandah, G.; Younis, K.; Khedher, M. 2008. Handwritten Arabic Character Recognition Using Multiple Classifiers Based on Letter Form. Proc 5th IASTED Int Conf on Signal Process, Pattern Recognit, & Appl, SPPRA 2008, 128–133.

Abandah, G.; Anssari, N. 2009. Novel Moment Features Extraction for Recognizing Handwritten Arabic Letters. J Comput Sci, 5(3): 226–232.

Abandah, G.; Khedher, M. 2009. Analysis of Handwritten Arabic Letters Using Selected Feature Extraction Techniques. Int J Comput Process Lang, 22(1): 49–73.

Arica, N.; Yarman-Vural, F. 2002. Optical Character Recognition for Cursive Handwriting. IEEE Trans Pattern Anal Mach Intell, 24(6): 801–813.

Burges, C. 1998. A Tutorial on Support Vector Machines for Pattern Recognition. Knowl Discov Data Min, 2(2): 1–43.

Dalal, S.; Malik, L. 2008. A Survey of Methods and Strategies for Feature Extraction in Handwritten Script Identification. Proc 1st Int Conf on Emerging Trends in Eng and Technol, 1164–1169.

Deb, K.; Pratap, A.; Agrawal, S.; Meyarivan, T. 2002. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. IEEE Trans Evolut Comput 6(2):182–197.

Deutsch, E. 1972. Thinning Algorithms on Rectangular, Hexagonal, and Triangular Arrays. Comm of the ACM, 15(9): 827–837.

Duda, R.; Hart, P.; Stork, D. 2001. Pattern Classification, 2nd edition, Wiley-Interscience, New York.

El Abed, H.; Margner, V. 2007. Comparison of Different Preprocessing and Feature Extraction Methods for Offline Recognition of Handwritten Arabic Words. Proc 9th Int Conf on Doc Anal and Recognit, ICDAR 2007, 974–978.

Freeman, H. 1961. On the Encoding of Arbitrary Geometric Configurations. IRE Trans Electron Comput, 10(2): 260–268.

Fukunaga, K. 1990, Introduction to Statistical Pattern Recognition, Academic Press Professional Inc., San Diego.

Guyon, I.; Elisseeff, A. 2003. An Introduction to Variable and Feature Selection. J Mach Learn Res, 3(1): 1157–1182.

Ha, T.; Bunke, H. 1997. Image Processing Methods for Document Image Analysis. In, Handbook of Character Recognition and Document Image Analysis, H. Bunke and P. Wang (editors), World Scientific, 1–47.

Hsu C-W, Lin C-J. 2002. A Comparison of Methods for Multi-Class Support Vector Machines. IEEE Trans Neural Netw, 13(2): 415–425.

Jain, R.; Kasturi, R.; Schunck, B. 1995. Machine Vision, MacGraw-Hill, New York.

Khedher, M.; Abandah, G. 2002. Arabic Character Recognition Using Approximate Stroke Sequence. Proc Workshop Arabic Lang Resources and Evaluation: Status and Prospects at 3rd Int Conf on Lang Resources and Evaluation, LREC 2002.

Kuhl, F.; Giardina, C. 1982. Elliptic Fourier features of a Closed Contour. Comput Graphs Image Process, 18(3): 236–258.

Liu, H.; Yu, L. 2005. Toward Integrating Feature Selection Algorithms for Classification and Clustering. IEEE Trans Knowl Data Eng, 17(4): 491–502.

Liu, C-L. 2008. Handwritten Chinese Character Recognition: Effects of Shape Normalization and Feature Extraction. Lecture Notes in Comput Sci, 4768: 104–128.

Lorigo, L.; Govindaraju, V. 2006. Offline Arabic Handwriting Recognition: A Survey. IEEE Trans Pattern Anal Mach Intell, 28(5): 712–724.

Märgner, V.; Pechwitz, M.; ElAbed, H. 2005. ICDAR 2005 Arabic Handwriting Recognition Competition. Proc Int Conf Doc Anal and Recognit, 70–74.

Märgner, V.; El-Abed, H. 2007. ICDAR 2007Arabic Handwriting Recognition Competition. Proc Int Conf Doc Anal and Recognit, 1274–1278.

Märgner, V.; El-Abed, H. 2009. ICDAR 2009 Arabic Handwriting Recognition Competition. Proc Int Conf Doc Anal and Recognit, 1383–1387.

Mezghani, N.; Mitiche, A.; Cheriet, M. 2002. On-line Recognition of Hand-written Arabic Characters Using a Kohonen Neural Network. Proc 8th Int Workshop on Front in Handwrit Recognit, 490–495.

Mitchell, T. 1997. Machine Learning, McGraw-Hill, New York.

Oliveira, L.; Sabourin, R.; Bortolozzi, F.; Suen, C. 2003. A Methodology for Feature Selection Using Multiobjective Genetic Algorithms for Handwritten Digit String Recognition. Int J Pattern Recognit Artif Intell, 17(6): 903–929.

Pechwitz, M.; Snoussi Maddouri, S.; Märgner, V.; Ellouze, N.; Amiri, H. 2002. IFN/ENIT–Database of Handwritten Arabic Words. Proc 7th Collque Int Francophone sur l'Ecrit et le Document, CIFED 2002, 129–136.

Pechwitz, M.; Maergner, V.; El Abed, H. 2006. Comparison of Two Different Feature Sets for Offline Recognition of Handwritten Arabic Words. Proc 10th Int Workshop on Front in Handwrit Recognit, 109–114.

Peng, H.; Long, F.; Ding, C. 2005. Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. IEEE Trans Pattern Anal Mach Intell, 27(8): 1226–1238.

Reiss, T. 1991. The Revised Fundamental Theorem of Moment Invariants. IEEE Trans Pattern Anal Mach Intell, 13(8): 830–834.

Rosenfeld, A.; Kak, A. 1976. Digital Picture Processing, Academic Press, New York.

Srinivas, N.; Deb, K. 1995. Multi-Objective Function Optimization Using Non-Dominated Sorting Genetic Algorithms. Evol Comput, 2(3): 221–248.

Stone, M. 1974. Cross-Validatory Choice and Assessment of Statistical Predictions. J R Stat Soc, 36(2): 111–147.

Theodoridis, S.; Koutroumbas, K. 2006. Pattern Recognition, 3rd edition, Academic Press.

Trier, O.; Jain, A.; Taxt, T. 1996. Feature Extraction Methods for Character Recognition: A Survey. Pattern Recognit, 29(4): 641–662.

Webb, A. 2002. Statistical Pattern Recognition, 2nd edition, Wiley, New York.

Wei, H-L.; Billings, S. 2007. Feature Subset Selection and Ranking for Data Dimensionality Reduction. IEEE Trans Pattern Anal Mach Intell, 29(1): 162–166.

Yu, L.; Liu, H. 2004. Efficient Feature Selection via Analysis of Relevance and Redundancy. J Mach Lear Res, 5(1): 1205–1224.

اختيار الميزات اللازمة للتعرف على الحروف العربية المكتوبة

ملخص: هناك العديد من طرق استخراج الميزات من الحروف المكتوبة بخط اليد، وهذه الطرق توفر مجموعات كبيرة من الميزات التي عادةً من تتضمن ميزات زائدة عن الحاجة وغير ذي صلة. نحتاج إلى عملية اختيار الميزات لاختيار مجموعة فرعية من الميزات تعطي دقة عالية في التعرف على الحروف ويمكن استخراجها بسرعة وكفاءة. في هذا البحث نستخدم تقنيات اختيار الميزات لتقييم مجموعة كبيرة من الميزات المستخرجة من الحروف العربية. حيث استخرجنا ٩٦ ميزة من ميزات هذه الحروف من جسم الحرف ومكوناته الثانوية وهيكله وحدوده. وقمنا بتقييم هذه الميزات واختيار أفضل مجموعات جزئية من أحجام مختلفة من هذه الميزات باستخدام خمس تقنيات لاختيار الميزات. هذه التقنيات تختلف في التعقيد من تقنيات انتقاء أفضل الميزات الفردية إلى الاختيار المتتابع الأمامي إلى خوارزمية التعظيم التطورية. ولقد وجدنا أن أفضل المجموعات الجزئية تتضمن ميزات المكونات الثانوية ووضع الحرف وواصفات فوربيه بيضاوي الشكل من الترتيب الأولي والعزوم وميزات حجم الحرف والميزات المستخرجة من حدود الحرف. وقمنا باستعمال ثلاثة أنظمة للتعرف على الأنماط لتقييم مجموعات الميزات المختارة من تقنيات الاختيار الخمس وذلك لإيجاد دقة التعرف على الحروف وعلاقة هذه الدقة مع حجم المجموعة. وجدنا أن من بين التقنيات الخمس كانت خوارزمية التعظيم التطورية هي الأبطأ في الاختيار ولكنها تختار مجموعات جزئية تعطي أعلى دقة تعرف. كما وجدنا أنه في معظم الحالات تحقق المجموعات الجزئية من حوالي ٢٠ ميزة تقريباً أفضل دقة تعرف.