Genetic-Neural Approach versus Classical Approach for Arabic Character Recognition Using Freeman Chain Features Extraction

Raed Abu Zitar
College of Information Technology, Philadelphia University, Jordan

Abstract: *This article presents a hybrid technique for the recognition of typed Arabic characters. Due to its curved and continuous nature, Arabic text has to go through words segmentation, character segmentation, feature extraction, and finally character recognition. In this work, Freeman Chain (FC) technique [20, 21] is used to generate a chain for every segmented character. This chain represents the extracted features. Moreover, two approaches are presented for the classification process. In the first approach, we use a classical sequential weighing algorithm that finds the closest available "Standard Character Template" to the extracted chain. In the second approach, we use Learning Vector Quantization (LVQ) (specifically LVQ3) technique for classifying the same chain. To improve the performance of that LVQ, the Genetic Algorithm (GA) [11, 23] is invoked for some additional training. We call our neural network with the GA "GALVQ3". For further robustness testing of both approaches, we add some artificial noise to the extracted chains and repeat simulations. In general, LVQ techniques provide higher classification rate even for cases where noise and partial observations exist. As a result, the GALVQ3 classifier is compact, online, robust, and feasible from hardware point of view.*

Keywords: *Arabic character recognition, neural networks, Freeman chain, feature extraction, LVQ.*

Received June 6, 2004; accepted October 4, 2004

1. Introduction

Arabic character recognition is one of the most challenging topics in character recognition. The fact that an Arabic character is written differently according to its position in the word makes its recognition a difficult task. For example, the Arabic letter below:

"ب"

Looks different if it is written at the beginning, at the middle, or even at the end of the word. Moreover, Arabic characters have different types and large number of curvatures, holes, and dots which make them even harder to recognize. Different applications nowadays require automatic and precise recognition for printed or hand written characters. Even personal applications require prompt identification of text as preliminary stage for subsequent crucial decisions. What we propose in this project is a method beyond handling well written, clear words and picking them up from a look up table. We propose a Genetic/Neural-Based system that can detect images with minimum effort and with fewer amount of data size. A system that has shorter processing time as it is based on parallel architectures. It also should be immune against unexpected levels of noise and tolerant against variations in input images. This system should have the attributes of an intelligent system that can make decisions under vague and uncertain observations. Moreover, it has to be compact and hardware realizable. The fact that Arabic letters have holes and dots within the characters requires special pre-processing techniques. These techniques include segmentation of pages into lines, segmentation of lines into words (sub-words), and words into characters. The characters then need to be coded in an efficient way utilizing the fact that what is being dealt with is only text and not all details of the image are really necessary. Previous techniques, such as horizontal and vertical projections have weakness in extracting some features of Arabic characters [1, 2].

This is due to the overlapping dots, and holes that increase segmentation error. In this work, we use a method called Freeman Chain (FC) to generate a chain coding (the feature) by detecting only the boundaries of the isolated character [35]. However, classification is implemented by two methods; one is basically based on a look-up table and the other is basically based on the application of artificial neural networks. The look up table method uses some intelligent search techniques based on frequency analysis that minimizes the number of iterations needed to search for a pattern. On the other hand, the second method uses neural networks. Neural networks are artificial intelligence paradigms that mimic neurons in living creature's brains [3, 18]. There are many types of neural networks, but what we are concerned about here is the classification ability based on extracted features. For instance, Self-Organizing Map (SOM) [25] is a suitable kind of neural networks for this application. It is an unsupervised learning technique that utilizes apparent and hidden features of some pattern to classify it into a unique class. It uses competitive learning techniques for the classification process. Adding another linear layer of supervised learning makes up an architecture equivalent to the LVQ, which is the network that we are using. The linear layer sharpens the classification between classes and helps in reducing errors.

However, if we switch back to the feature extraction method that we are using, which is FC, we find that although it can be modified to handle noise added to the letters like alterations in fonts, extra ink, lack of ink, or variations in thickness, it is still sensitive to letter size or different curvatures in the characters. This problem can be overcome by using normalization and enhancement added to that technique.

With the addition of neural networks we will have a classifier that can tolerate the effects of different types of noise. In this paper we will deal with well-typed letters using some standard and fixed font. In future we hope that we will be able to extend it to hand-written characters. We should keep in mind that recognition of hand-written characters is a real challenging problem. The diversity of styles makes it difficult to unify the extracted features of some characters in one class. In all cases, it is a well-known fact that no matter what is the classification technique, the segmentation method has to be reliable, especially for characters. At the classification stage, it is required to develop more efficient feature extraction technique that can discover similarities among similar characters typed under different conditions, and still distinguish them as a unique class. As a matter of fact, the more successful the feature extraction stage, the easier the classification process will be [36]. With proper extraction methods, the dots and holes problem could be converted into assisting factor in the classification process. Neural networks have shown great success in classification applications even if the input features were noisy and deceiving [33, 37, 38].
Adequate research has been accomplished with Arabic character recognition.

Amin A. *et al.* [35] suggested methods based on the morphology of the printed Arabic text, using vertical histograms for segmentation. Refat Ramsis *et al.* [2] used threshold criterion in segmentation of characters with Accumulation Invariant Moment (AIM) in building classifiers. Others used free-like decision trees on extracted features that covered number of points in the character, the relative position of dots to the main stroke, and the number of secondary strokes and slopes of secondary components [39]. Next, we will present our segmentation technique of the words and then we will present our approach for isolating characters before we extract their features with FC. In later stage, those features are used in the classification using both our classical and the GALVQ3 methods. In 1961, Freeman H. [20, 21] proposed and implemented a scheme in order to encode efficiently any contour-like map. The first assumption is that for any digitized binary picture, each pixel has eight neighboring pixels at the outside boundary. The coding method is to select from any closed curve any pixel at the outside boundary, then to move counter clock wise around the boundary of the curve extracting the neighboring direction of each pixel moved to. At each step the neighboring direction of each pixel is registered to form successively a chain of directions named as FC. The directions, that are shown in Figure 1-a, are given numbering as follows; 0 for upper left, 1 for left, 2 for lower left, 3 for lower, 4 for lower right, 5 for right, 6 for upper right, and 7 for upper. Figure 1-b shows an example of a binary picture and its associated FC.

2. Segmentation and Chains Extraction

The first step is the line segmentation of the text. The TIF file that contains the Arabic text is processed here by first separating lines from each other according to some threshold. Threshold has been determined by experiments. After separating the TIF Line (TL), the line can be processed by FC method. The FC works by starting at the first black pixel from left to right beginning with first blank line to last blank line. If a black pixel is found, Freeman counter-tracing-mechanism starts to move around the body of segment quantizing, at each step, the direction of the next neighboring point. This continues until moving around mechanism reaches the starting pixel. This process ends up with an FC of a word or sub-word. The next step is implementing the following successively:

1. Partial Character Segment Formation (PCS) which basically separates frames of characters.
2. Freeman chain extraction for an isolated character.
3. Relative position of the rectangular frame containing PCS.

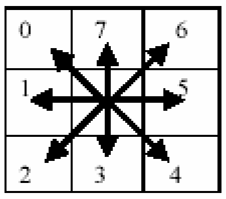
FC chains we are looking for are actually the partial segments of the separated characters (PCS's). The FC method requires much less storage and processing time than any segmentation technique since we detect only the borders of the characters. An example of TL segmentation is shown in Figure 2. It is a text containing two graphical Text Line segments (TL's). First TL contains 14 PCS's (a PCS could be a character, dot, hole, part of a word, word, or any isolated frame of continuous characters), each PCS gives a FC and is enclosed by a small rectangle whose upper left corner and right bottom corner are also computed from pixels numbers according to location on screen (see Textual Line 1). It is worthy to notice that the overlapping problem, the dots and holes problem are taken care by using our technique. The dots and holes will have their own chains, while any two overlapped characters will have their own FC's as long as they are separated by blank. Figure 3 shows examples of a single word consisting of 5 FC's (and not PCS's); FC1, FC2, and FC3 are regular characters, in addition to a dot (FC4), a hole (FC5). Here, there are two overlapping characters (between FC3 and FC1) (i. e., their vertical projections overlap). Other segmentation techniques such as the horizontal-vertical [30] projections do not handle this problem properly as the projections usually do not reflect separations between characters. Smoothing technique can be implemented to enhance the features of the extracted chain. Any bent in the body of the character can be ignored if some odd or minor numbers for directions appeared in the sequence of the chain.

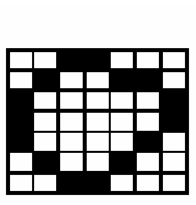
The Arabic characters are converted into straight vertical or horizontal lines. The heuristics we used are the result of many experiments on Arabic characters. These heuristics can be considered as general rules for almost any type and style of any set of characters.
Finally, the extracted FC's are re-coded in pairs of (direction, length). This process shortens the representation and provides a better view at the FC, besides having much less storage and processing time.

As mentioned earlier, the process of isolating the characters and finding their FC's comes after having separated the PCS for every TL. However, the FC of a word or sub-word is first found, the PCS are generated, and then FC's for every single isolated character (within the PCS) are computed. To implement character isolation, that part of the chain with long straight lines in any of the horizontal directions (1 for left and 5 for right) is traced. These two directions lie in the direction of the baseline and form the place of characters concatenation. Character isolation execution will depend on the presence of long sets of opposite 1 and 5 directions. To implement character isolation, two conditions must be satisfied:

1. They must be long enough to indicate the end of a single character
2. Their positions must be near the base line within 3 pixels.

If these two conditions were satisfied, character isolation is implemented.





(a) (b)

Figure 1. Freeman chains and their directional encoding.

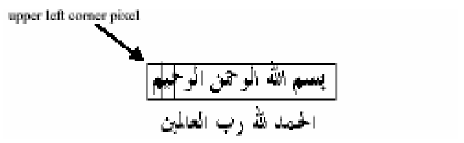


Figure 2. A Sample graphical text.

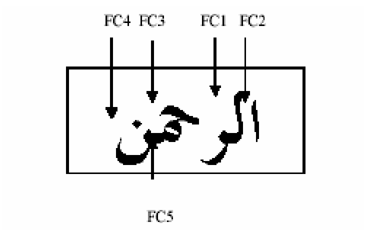


Figure 3. Overlapped PCS's with non-overlapping FC's.

3. The Problem of Dots and Holes

Before showing an example of isolating and constructing FC's, we have to explain the process of chains extraction of dots and holes that exists extensively in Arabic words. Dots and holes are important features within the body of their own characters. They must be identified and related to their own characters. This will be done through the use of two descriptive and quantitative parameters listed in word segmentation. These are the upper left corner and the lower right corner of the rectangular box enclosing the FC of the dot or of the hole under consideration. These positions are to be compared with the respective positions of other non-dots FC's so they lie on their ranges. For example, the FC of the dot corresponding to the first word of Figure 2:

Have the positions (55,613), (65,624) which are very related to positions (26,538), (72,627) of the FC of the same word without a dot. The position of the dot s below the lower right corner of that word which implies that this dot belongs to the first character of the word and that will be used in the classification of that character. Note that dots do not come by themselves, the are parts of other characters.

The following is an example of FC encoding and smoothing enhancement for an Arabic word that forms by itself a PC (since it contains a frame of Arabic characters).

The FC (without dots) for the Arabic word:

is given by:

23233233245435443344334334332111111111110112
22333344455555555555555555555557456456555555
55555555555765656554433555455564555555555555
56555551076777777777776711077677011123332343
54354354432112011111111211012110107766577777
76000011001111111121212233335544544333221111
11100777777707770111133322344443444434321111
1111111110077777706770707077007711

The smoothed and paired FC of the same word (without dots) Paired Freeman Chain (PFC) (direction, length) is:

2(1)3(13)4(8)3(3)1(16)3(10)5(94)7(24)1(4)3(14)1(23)
7(21)1(19)3(6)5(6)3(9)1(10)
7(14)1(4)3(14)1(16)7(23)1(2).

Notice that the fived FC's always starts with 5 (left to right) as a standard for the start of chain extraction for the isolated characters. If the PFC of this word is fractured into isolated characters to form Fived Paired Freeman Chains (FPFC's), then we will have 4 FPFC's as follows:

FPFC of "ن" without dot:

5(27)7(24)1(4)3(14)1(23)3(10)

FPFC of "ع":

5(23)7(31)1(19)3(6)5(6)3(9)1(10)3(16)

FPFC of "ب" without dot:

5(20)7(30)1(4)3(14)1(16)3(16)

FPFC of "لـ":

5(24)7(39)1(2)2(1)3(13)4(8)3(3)1(16)3(14)

Keep in mind that the FPFC's belong to isolated characters with chains formation starting from left to right (the 5 direction) opposite to the direction of writing in Arabic. As mentioned earlier, the FPFC's are accompanied with two numbered position numbers for the upper left pixel and the lower right pixel of the rectangle containing the isolated character. This part will be declared clearly in simulations.

4. Our Classical Classification Algorithm

The last step in the proposed character recognition is the classification. Every isolated character (unknown FPFC) now is recognized according to the following algorithm:

1. Access the existing set of references of FPFC's (call it R_i). Look for matches with the sequence of directions (ignoring the length of direction). Established a matching list of all R_i references that does match.
2. Repeat next step (labeled a) number of times equal to number of directions in the unknown FPFC: (a) The value of each length number for current direction in the unknown FPFC is quantized module 10 as an r value. Then a scan for its presence with that range is compared with the references of the established matching list of the FPFC R_i. If matching occurs then a counter (*counter* (r)) for that range for that reference is incremented.
3. A weighted sum w_{si} is obtained for every reference FPFC by multiplying the sums of step (2a) by their r values so we will have

$$w_{si} = w_{si} + r \,.\, count\,(r).$$

4. The resulted w_{si} are sorted in descending order. And the highest reference is taken as the decision class.
5. In case, of similar FPFC's dots and holes are used for classification decision with simple *if-then* statements.

Table 1. All possible Arabic characters with their codes.

Character	Code	Character	Code	Character	Code
ا , ـا	0	فـ	18	تـ , ـتـ	36
ط , ـط	1	ـفـ	19	بـ , ـبـ	37
بـ , ـبـ	2	ف , ـف	20	ذ , ـذ	38
ـب	3	مـ , ـمـ	21	ز	39
د , ـد	4	م , ـم	22	ض	40
ر , ـر	5	ه	23	خـ	41
صـ , ص , ـصـ , ـص	6	ـه	24	جـ	42
حـ , ـحـ	7	ـهـ	25	ح	43
ح , ـح	8	ن , ـن	26	خ	44
قـ	9	سـ , ـسـ	27	غـ	45
ق , ـق	10	س , ـس	28	غ	46
ـقـ	11	ى , ـى	29	ـغـ	47
عـ	12	ء	30	ـغ	48
ع	13	ک	31	ت	49
ـع	14	ك , ـك	32	ي	50
ـعـ	15	و , ـو	33	شـ , ـشـ	51
لـ , ـلـ	16	ظ , ـظ	34	ش , ـش	52
ل , ـل	17	نـ , ـنـ	35		53

5. Simulations and Results

The recognition system and its algorithm outlined above have been tested through a number of textual inputs. The system proves to reconstruct almost more than 90% of the characters in each textual line. However, there are some misrecognition states especially with those complicated characters such as the character:

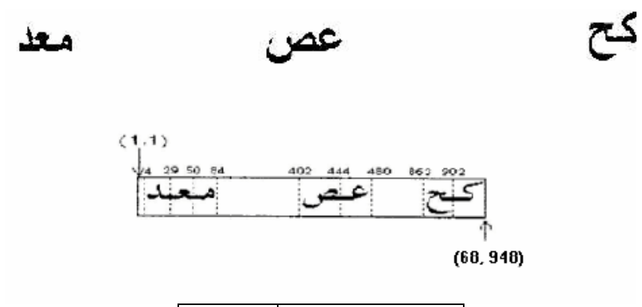
طـ

Each character, here, hasbeen given a special code, see Table 1. The result of the classification is the code of the character and its associated position (upper left and lower right corners) on the screen. Table 1 shows the possible 53 character shapes in Arabic and their codes used in our simulations. The following experimented textual lines and their character recognition results are shown. Please, note that we are using the codes shown in Table 1. Also, the positions shown in the output table for each textual line indicate the upper left and the lower right coordinates of the pixels that form a rectangular boundary around the isolated character. The "*" sign indicates a misclassification.

5.1. Textual Line 1

This line contains the following three words. Each consists of two or three characters.

The system has recognized all seven characters. It outputs every identified character as output code in addition to the four relative positions of the character.



Code	Positions
31	(1, 902) (68, 948)
8	(1, 863) (68, 902)
21	(6, 50) (39, 84)
15	(6, 29) (39, 50)
4	(6, 4) (39, 29)
12	(9, 444) (57, 480)
6	(9, 402) (57, 444)

5.2. Textual Line 2

عمر حصد مهر

In this line, the system has recognized 7 out of 9 characters. The output as follows:

Code	Positions
7	(1, 468) (35, 505)
6	(1, 428) (35, 468)
4.16	(1, 390) (35, 410)
21	(5, 58) (52, 101)
28 *	(5, 2) (52, 58)
12	(5, 868) (51, 906)
21	(5, 839) (51, 868)
5	(5, 799) (51, 839)

(*) Stands for misrecognized character. It is noticed that the misrecognition comes from the multi holes character and any neighboring character from the small group of characters.

The same misrecognition situation happens here where the small characters have been mixed with their neighboring characters. Small characters need to be supplied with further parameters to enhance their classification accuracy. These parameters may include their relative positions to the base line and the size of their enclosing box.

The FC approach solves the problem of dots and holes in addition to the overlapping problem between two different segments and words. This is because FC technique traces segments along their boundaries and they are separate as long as one white pixel separate them. Our approach solves the problem of aligning the dots and holes relative to their original characters by recording the coordinate of each segment whether it is a dot, a hole, or a real segment. This technique proved to be useful in the recognition stage where the size of the enclosing box was used to identify the one dot, two dots, large hole and small hole. Also, the size can be used in enhancing the recognition of the character “Ç “ (in English: Alef). This character has a relatively long height (more than 30 pixels) and small width (less than 16). One problem we have is that there are still some difficulties in the character isolation process within a segment, specially with group of small characters such as:

“د “, “ر “,” and “ و “

Moreover, the technique used in recognizing dots, holes and “|“ might be extended to include the positioning of the isolated characters relative to the base line. This condition will isolate those characters that lie below the base line such as:

“و“, “ن “, “ح “, “ ر “, “ ز “

In a separate group from those that lies above base line such as:

“ك “, “ب”, “ف “, “ق”

Consequently, the use of the enclosing rectangular box and its dimension will enhance the character recognition process. Some characters have more height than width such as:

“ح” and “ك”

With the previously mentioned technique we may isolate them in a separate group. Other close characters such as:

“د “ and “ ل “

May be isolated with such techniques. However, these added conditions and methods require more experimentation and “isolation threshold” determination.

Our system has recognized typed Arabic characters with rate more than 90%. The remaining 10% is due to many factors such as noise accumulated by the scanner and sensitivity. In addition to the curves, dots, holes, and overlapping of the Arabic characters that makes the isolation stage more difficult than other stages.

Experiments with the last three text lines gave better recognition rate than previous ones. We used higher quality scanners, and repeated the usage of this classical algorithm until less error rates were achieved. Those experiments show recognition rates of 94%. We believe that if better segmentation techniques were used to avoid the problem of overlapping between characters, higher recognition rates may be achieved. For example, if we look at the words below:

لإ or لي

It is clear that there is some overlapping between the characters of the same word. Much more enhanced character segmentation is required to avoid such problem.

From the previous simulations we may conclude that the proposed system can be used for other languages that are not cursive such as English and is expected to give higher recognition rates. Our method is general and can be implemented on any single character set whatever the style, font, and size of that set.

Figure 4 shows a flow chart that summarizes the whole process of segmentation, isolation and classification for the classical recognition system.

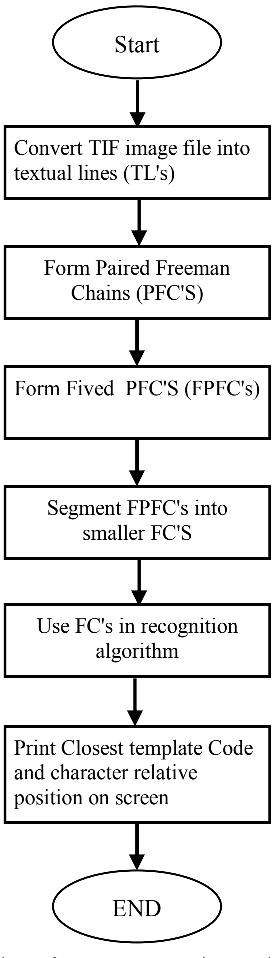

Start
Convert TIF image file into textual lines (TL's)
Form Paired Freeman Chains (PFC'S)
Form Fived PFC'S (FPFC's)
Segment FPFC's into smaller FC'S
Use FC's in recognition algorithm
Print Closest template Code and character relative position on screen
END

Figure 4. Outline of text segmentation and classification.

6. Neural Network Recognition Approach

The Neural Network (NN) is a fast, parallel, and compact approach for processing the extracted features of the isolated characters. It has the potential of being implemented with hardware [14]. This could be a major advantage against digital computer-based classification algorithms. The Self Organizing Map (SOM), which could be viewed as the first stage of LVQ, does internal representation of the data classifying them into different corners of the template space. A second stage (a linear activation and supervised stage) minimizes classification error by fine tuning the clustering boundaries between clusters of different classes.

Figure 5 shows diagram of an LVQ neural network architecture. The ability of neural networks to overcome noise and tolerate short, odd and abrupt fluctuations in input data makes it always preferable for recognition tasks. For the purpose of fair comparisons between what we called our classical method and the artificial intelligence inspired methods (i. e., NN), we used the same extracted features for the same examples in testing an SOM classifier. Our neural-based method is based on the LVQ3 neural network. The first stage of it consists of an SOM. The SOM was originally proposed by Kohonen [15, 25]. It is an unsupervised learning technique that has the ability to learn from examples and extract statistical properties of the examples presented during training. The input itself is used to stimulate the SOM to classify it into classes according to embedded features within input vector itself. In a way, this feature map is analogous to the spatial organization of sensory processing areas in the brain. For instance, let $m_i(t)$ be denoted as the weights of the ith neuron in an SOM during time instant t, the weights of SOM then are updated according to the following simple formula:

$$m_i\,(t + 1) = m_i\,(t) + h_{ci}\,(t)\,[x\,(t) - m_i\,(t)] \qquad (1)$$

$h_{\mathrm{ci}}(t)$ is the so called *neighborhood kernel*, which determines the size of the neighborhood of the ith neuron within which are all neighboring neurons will be updated in response to the present feature vector $x(t)$. Initially, the neighborhood is large. As training goes on, the size of the neighborhood shrinks. This process is called "clustering" process.

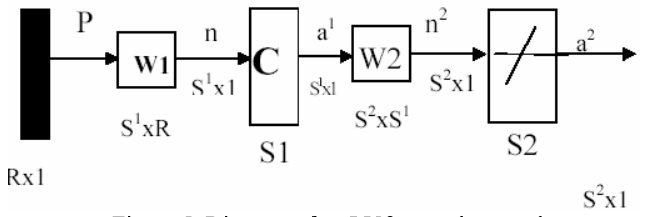


Figure 5. Diagram of an LVQ neural network.

Clustering converges when no neighboring neurons get their weights updated. With the LVQ, in general, for some input vector $x(t)$ a code y_c is formed such that $c = arg\ min_i\ \{\|\ x - y_i\ \|\}$. Specifically, for LVQ3 method, that we used, Kohonen [19, 25] suggests that if y_i and y_j are weights of neurons associated with different classes then y_i and y_j code words are updated according to the following:

$$y_i\,(t + 1) = y_i\,(t) - k\,(t)\,[x\,(t) - y_i\,(t)] \qquad (2)$$
$$y_j\,(t + 1) = y_j\,(t) + k\,(t)\,[x\,(t) - y_j\,(t)] \qquad (3)$$

On the other hand, if both y_i and y_j belong to the same class as x(t), and x(t) fall in a window centered at the cluster boundary of these two classes, then

$$y_k\,(t + 1) = y_k\,(t) + e\,.\,k\,(t)\,[x\,(t) - y_k\,(t)],\quad k \text{ belongs to } \{i, j\},\ 0.1 < e < 0.5 \qquad (4)$$

where k(t) is time-varying learning rate, and the optimal value of e depends on the size of the window, being smaller for narrower windows. This algorithm is self stabilizing and optimal placement of the yi does not change in continual training.

The learning process starts using the above mentioned LVQ3 technique. For this application, we started by creating 53 templates for all possible character shapes of font 12 and regular type writing. These templates are the reference templates. The position of each isolated character is found but is not included in template. We used an SOM of (4x20) grids. In that sense, we had enough neurons to cover the possible 53 classes with 27 classes more. The extra 27 classes give more freedom for the SOM in distributing its output all over the space having more separations between adjacent neurons. In addition to that, we used a linear-activation layer of 53 neurons where each neuron is associated with a code number. This code number is the target for that linear supervised layer. By the end of successful training phases, only one distinct neuron will go on when the LVQ3 is subjected to a specific Fived-Paired-Freeman-Chains (FPFC) input. Each neuron (at output layer) represents a class (out of 53). It should be noted that the SOM classifies input vectors into sub-classes, while the supervised layer takes the output of the SOM and combines those subclasses into one class. If it is happened, and a neuron went on for more than one class, learning continues until a total one-to-one mapping between a set of 53 neurons and the input space of the reference FPFC's is established. Initially the LVQ3 algorithm is used in training. However, the Genetic Algorithm (GA) is called to take over training of the supervised layer (the output layer) when the error (mean square error) of the LVQ3 algorithm reaches a flat neighborhood. The LVQ3 is turned off and the GA presumes the learning process until the error level reaches the preset criterion. For many FPFC's references, the output of LVQ3 reached the preset criterion. However, when all the training set (53 codes) is used in one patch, the GA is needed to finish the training job. In that case, the LVQ3 could not be without the help of the GA. It should be noted that the output layer is a single layer supervised neural network and a global training algorithm such as the GA is needed to escape local minima. The GA in its simplest form consists of three basic operations: reproduction, crossover, and mutation [11, 23]. The basic building block in the GA is the "string"; which is a sequence of bits representing variables of the search space. The bits themselves form the genotype and their decoded values are the phenotype. All GA operations are implemented over a finite population of strings. As search process goes on, the average fitness for the population of strings is expected to increase. Fitness is measured using some objective functions that are related to the criterion needed to be optimized. Figure 7 shows the blocks of the basic operations of the standard GA.

The criteria that the GA optimizes here is the RMS (Root Mean Square) error between the targets that are the assigned labels for every class and the binary input patterns that represents the subclasses generated by the SOM layer (inside the LVQ3) network. As indicated earlier, the LVQ3 network is actually an SOM network followed by some linear activation supervised layer. The GA trains only the supervised layer which is the output layer for the total GALVQ3 network (the name of our NN). The SOM network had 80 neurons, and the supervised layer had 53 neurons (equal to the number of classes). The learning rate of the SOM started with a value of 1.0 and kept going down until reached value close to 0.01. The GA had a population of 200 strings, probability of crossover of 0.6, and probability of mutation of 0.01. However, when the standard deviation of the RMS error showed signs of mature convergence, the probabilities of mutation and crossover were modified considerably. This was done to stir the pool of the strings toward new more recombination. We used a deterministic criterion of selection in the reproduction phase of the GA [11, 23]. Figure 6 shows GALVQ3 (LVQ3 GA supported) learning error versus learning cycles. The plots of the total RMS error for both techniques show how the GALVQ3 method manages to reach lower error values for the same problem. The GALVQ3 is a heuristic algorithm that contains some random operators implemented by the GA. That is why its plot has a fluctuating nature compared to the only LVQ3 plot. The GA in the GALVQ3 kept disturbing the saturation of the error levels in a manner that pushed the search process to move toward lower levels of error. After completion of the learning stage for the 53 reference FPFC's, we repeated the simulations done on textual line 1, textual line 2, textual line 3, textual line 4, textual line 5, textual line 6, textual line 7. Moreover, experiments were done on 500 characters picked randomly from different resources. Out of around 500 characters, the GALVQ3 recognized around 485 making up 97% classification rate. The input to the GALVQ3 was the FPFC of the isolated character followed by any FPFC's for dots or holes part of that character. One bit of 1 is also added if the dot was up, or a bit of 0 is added if the dot was down. The output of the GALVQ3 was only the number of the only active neuron (numbered from 1 to 53) triggered by the input FPFC. Those neurons are attached to the constant characters codes that are shown in Table 1. The major problem with the GALVQ3 is that it requires a fixed length input vector for all FPFC's used as input. We had to assume a maximum length of 40 for every FPFC. If the actual length of the FPFC was less than 40, the FPFC was padded by 0's until length becomes 40. This padding process was implemented during learning and retrieval.

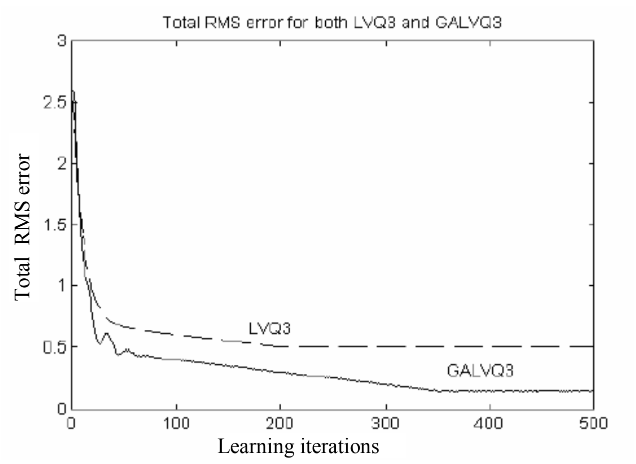


Figure 6. The RMS error plots for both LVQ3 and GALVQ3.

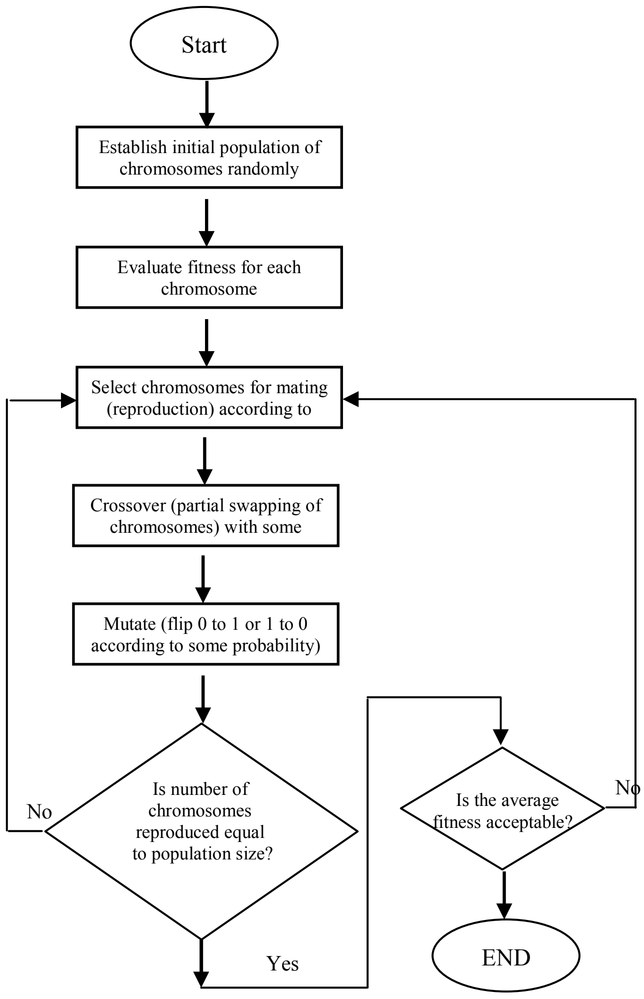

Start
Establish initial population of chromosomes randomly
Evaluate fitness for each chromosome
Select chromosomes for mating (reproduction) according to
Crossover (partial swapping of chromosomes) with some
Mutate (flip 0 to 1 or 1 to 0 according to some probability)
Is number of chromosomes reproduced equal to population size?
No
Yes
Is the average fitness acceptable?
No
END

Figure 7. Basic standard GA operations.

7. Conclusions and Discussions

This character recognition system as a whole is compared to other systems. Amin proposed several systems. The best Character Recognition rate (CR) achieved was 95.4% [31]. Badi [32] used structural features, just as [31]. However, unlike Amin, who used a nearest-neighbor classification technique, Badi used a decision tree technique. Badi's system gave a CR of 90%. El-Wakil in [10] applied his mechanism to isolated characters utilized structural features in a chain code, but unlike our work, used a nearest-neighbor method for classification. His method yielded a CR (identical to ours) of 93%. The best CR claimed for any work in on-line Arabic Character Recognition is the 99.6% stated in [29]. This work also used structural features and a decision tree for classification. The system was applied to the recognition of characters as well as mathematical formulae [18]. Others used Fourier-based descriptors [16, 17, 24] to precisely reflect the curved nature of Arabic characters and got good results of around 95.5% recognition rate. In summary, we can also claim that the (preliminary) CR rate achieved with our system is among the best reported, but still can be improved significantly. However, it is crucial that we reach that high CR rate without sacrificing too much the simplicity of the system, nor indeed, its real time nature.

In summary, we have demonstrated in this work a comprehensive comparison between two methods for Arabic character recognition. These two methods can be extended to any type of characters. The character recognition is first implemented using a classical approach that uses a deterministic weighing algorithm for selecting the closest template according to a FC feature. The second approach uses an LVQ3 neural network assisted with Genetic Algorithm (GA). We called the second approach "GALVQ3". The two approaches do only the classification process while the features extraction mechanism is similar for both of them. Testing with noisy inputs is implemented with both approaches. Four types of noise were added independently to each of the 53 features templates and the noisy templates were fed to the two classifiers. The neural network classifier (GALVQ3) has outperformed the classical approach classifier described in section 4. The GALVQ3 has a self organizing ability to respond to different inputs and to converge toward a unique class. The "supervised" layer that comes after the SOM layer in the GALVQ3 is trained with the GA. The GA is known as a global optimizer that has the ability to "dig" deep into the search space of the weights of the supervised layer and discover optimal solutions. The SOM layer in the GALVQ3 self-organizes its output into subclasses. The supervised layer in the GALVQ3 wraps up associated subclasses into one distinct and labeled class. Training the supervised layer with GA is proved to be faster and better than training it with gradient descent methods. That is depicted in results shown in this work. On the other hand, typed Arabic characters are still special due to the fact that they have curvy nature and there are more holes and dots in this language than any other major language. With FC method, the curves of its characters can be converted into an informative sequence of numbers. By enhancing the segmentation process and further repetition of training for the GALVQ3, it was possible to increase the correct recognition. Supplementary features may be used, such as Zernike Moments [27] can be added, or using different encoding, translation, scaling, and rotations will help in providing richer feature vectors for isolated characters [34]. Future work will definitely include using hybrids of these methods in the on-line recognition of hand written Arabic symbols [28]. This is our ultimate goal, as most recent research is focused more on hand written characters rather than simply printed ones.

Finally, the character recognition part is an important step toward automating documents processing and providing time-critical information. AI techniques such as neural networks show a great potential in future industrial applications that support information capturing and document analysis and recognition. It is not going to be long before an intelligent "characters-recognizer" is used as supplementary part in banks, post offices, hospitals, and government agencies.

References

[1] Abu Zitar R. and Al-Fahed A. M., "Performance Evaluation of Genetic Algorithms and Evolutionary Programming in Optimization and Machine Learning," *Cybernetics and System an International Journal*, vol. 33, no. 1, 2002.

[2] Abu Zitar R. and Hassoun M. H., "Neurocontrollers Trained with Rule Extracted by a Genetic Assisted Reinforcement Learning System," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 859-879, July 1995.

[3] Amin A. and Mari J. F., "Machine Recognition and Correction of Printed Arabic Text," *IEEE Transactions on Systems Man and Cybernetics*, vol. 19, no. 5, September 1989.

[4] Amin A., Kaced A., Haton J., and Mohr R., "Hand Written Arabic Character Recognition by the I.R.A.C. System," *in Proceedings of the 5th International Conference on Character Recognition*, Miami, FL, pp. 729-731, 1980.

[5] Badi K. and Shimura M., "Machine Recognition of Arabic Cursive Scripts," *Transactions of the Institute of Electronics & Communications Engineers*, Japan, vol. 15, 1982.

[6] Bailey R. and Srinath M., "Orthogonal Moment Features for Use with Parametric and Non-Parametric Classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, 1996.

[7] Belkasim S. O., Shridhar M., and Ahmadi M., "Shape Recognition Using Zernike Moment Invariants," *in Proceedings of the Asilomar Conference on Circuits*, vol. 1, pp. 161-171, 1989.

[8] Belkasim S. O., Shridhar M., and Ahmadi M., "Pattern Recognition with Moment Invariants: A Comparative Study and New Results," *Pattern Recognition*, vol. 24, no. 12, pp. 1117-1138, 1991.

[9] Bouhlila K., Hamrouni M. K., and Ellouze N., "Method of Segmentation of Arabic Text Image into Characters," *in Proceedings of the First Kuwait Computer Conference*, Conference Proceedings, pp. 442-446, March 1989.

[10] Carver M., *Analog VLSI and Neural Systems*, Addison-Wesley Publishing Company, 1990.

[11] Davis L., *Handbook of Genetic Algorithms*, Van Nostrad, Reinhold, New York, 1991.

[12] Desai M. and Cheng H., "Pattern Recognition by Local Radial Moments," *in Proceedings of the International Conference on Pattern Recognition*, pp. 168-172, 1994.

[13] DiZenzo S., DelBuono M., Meucci M., and Spirito A., "Optical Recognition of Hand-Printed Characters of any Size, Position, and Orientation," *IBM Journal of Research and Development*, vol. 36, no. 3, pp. 487-501, 1992.

[14] El-Desouky A., Salem M., and Arafat H., "A Handwritten Arabic Character Recognition Technique for Machine Reader," *International Journal for Mini Microcomputer*, vol. 14, no. 2, pp. 57-61, 1992.

[15] El-Emami S., "Machine Learning of Handwritten and Type Written Arabic Characters," *PhD Thesis*, University of Reading, UK, 1988.

[16] El-Sheikh T. S., "Recognition of Handwritten Arabic Mathematical Formulas," *in Proceedings of the UK IT 1990 Conference*, Southampton, London, UK, pp. 344-351, 1990.

[17] El-Sheikh T. S. and El-Taweel S. G., "Real-time Arabic Handwritten Character Recognition," *in Proceedings of the 3rd International Conference on Image Processing and its Applications*, IEE. London, UK, pp. 212-216 1989.

[18] El-Wakil M. S. and Shoukry A. A., "On-line Recognition of Hand Written Isolated Characters," *Pattern Recognition*, vol. 22, no. 2, pp. 97-105, 1989.

[19] Eisenstein B. and Vaccaro R., "Feature Extraction by System Identification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC 12, no. 1, pp. 42-50, 1982.

[20] Freeman H., *Analysis of Line Drawings, Digital, Image Processing and Analysis*, NATO Summer School, in Simon J. S. and Rosenfeld A. (Eds), Nordof, 1977.

[21] Freeman H., "On the Encoding of Arbitrary Geometric Configuration," *IRE Transactions Electronic Computers*, no. 2, pp. 260-268, 1961.

[22] Fukushima K. and Wake N., "Handwritten Alphanumeric Character Recognition by the Neocognitron," *IEEE Transactions Neural Networks*, vol. 2, no. 3, pp. 355-365, 1991.

[23] Goldberg D. E., *Genetic Algorithms in Search Optimization, and Machine Learning*, Reading, MA, Addison-Willey, 1989.

[24] Granlund G. H., "Fourier Preprocessing for Hand Print Character Recognition," *IEEE Transactions on Computers*, February 1972.

[25] Hassoun M. H., *Fundamentals of Artificial Neural Networks*, MIT Press, Cambridge, 1995.

[26] Jefferies C., *Code Recognition and Set Selection with Neural Networks*, Boston, Birkhauser, 1991.

[27] Kauppinen H., Seppanen T., and Pietikainen M., "An Experimental Comparison of Autoregressive and Fourier-Based Descriptors in 2D Shape Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 201-207, 1995.

[28] Khotanzad A. and Hong Y. H., "Invariant Image Recognition by Zernike Moments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 5, 1990.

[29] Kohonen T., *Self Organization and Associative Memory*, Series in Information Sciences 8, Springer-Verlag, 1983.

[30] Kohonen T., "Self-organizing Maps: Optimization Approaches," in Kohonen T., Makisara K., Simula O., and Kanga J. (Eds), *Artificial Neural Networks, IEEE*, New York, pp. 1147-1156, 1996.

[31] Kosko B., "Neural Networks for Signal Processing," *PhD Dissertation*, University of Reading, UK, Englewood Cliffs, NJ, Characters, 1988.

[32] Nishimura M. and Van der Spiegel J., "Pattern Recognition Based on Orientation and Linestops Using an Orientation Sensor and Multilayered. Neural Network," *in Proceedings of SPIE'95*, 1995.

[33] Perantonis S. J. and Lisboa P J. G., "Translation, Rotation, and Scale Invariant Pattern Recognition by High-Order Neural Networks and Moment Classifiers," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, 1992.

[34] Persoon E., "Shape Discrimination Using Fourier Descriptors," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC 7, no. 3, 1977.

[35] Ramsis R., El-Dabi S., and Kam A., "Arabic Character Recognition System: A Statistical Approach for Recognizing Cursive Typewritten Text," *Pattern Recognition*, vol. 23, no. 5, pp. 485-495, 1990.

[36] Rumelhart D. E., McClelland J. L., and the PDP Research Group, *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, vol. 1, MIT Press, Cambridge, Mass., 1986.

[37] Sanossian H. Y., "An Arabic Character Recognition System Using Neural Network," *in Proceedings of the IEE Workshop on Neural Networks for Signal Processing*, pp. 340-348. 1996.

[38] Simon J. C., "Uncertainty versus Computational Complexity," in Johnson J. H., McKee S., and Vella (Eds), *Artificial Intelligence in Mathematics*, Oxford University Press, 1994.

[39] Wang D. and Xie W., "Invariant Image Recognition by Neural Networks and Modified Moment Invariants," *in Proceedings of SPIE'96*, 1996.

Raed Abu Zitar obtained his BSc in electrical engineering from the University of Jordan in 1988, his MSc in computer engineering from North Carolina A&T State Universityin 1989, and his PhD in computer engineering from Wayne State University Detroit, Michigan, USA in 1993. Currently he is an associate professor at the Department of Software Engineering, Faculty of IT and in the capacity of the Dean of Scientific Research and Graduate Studies, Philadelphia University, Amman, Jordan. His research interests are in neural networks, signal processing, robotics, genetic algorithms, and machine learning.