Journal of Computer Science 4 (9): 744-751, 2008
ISSN 1549-3636
© 2008 Science Publications

Back Propagation Neural Network Arabic Characters Classification Module Utilizing Microsoft Word

Hamza, Ali A.
Faculty of Computer Science and Information Technology,
Al- Isra Private University, P.O. Box 22, Code 11622, Amman, Jordan

Abstract: Problem statement: Arabic character recognition has been one of the last major languages to receive attention. This may be attributed to the inherent complexity of both printed and handwritten Arabic characters. The objectives of this study were to: (i) summarize the main characteristics of Arabic language writing style. (ii) suggest a neural network recognition circuit. **Approach:** A Neural network with back propagation training mechanism for classification was designed and trained to recognize any set of character combinations, sizes or fonts used in Microsoft word. **Results:** The proposed network recognition behaviours were compared with perceptron-like net that combines perceptron with ADALINE features. These circuits were tested for three character sets combinations; 28 basic Arabic characters plus 10 numerals set, 52 Latin characters and 10 numerals only. **Conclusions:** The method was robust and flexible and can be easily extended to any character set. The network exhibited recognition rates approaching 100% with reasonable noise tolerance.

Key words: Pattern recognition, classification, Artificial Neural Networks, back propagation, character recognition

INTRODUCTION

Arabic script with its basic character shapes is adapted for writing in many languages such as Persian, Urdu, Malay, Kurdish and Sindhi. Considerable research has been done recently on Arabic character and text recognitions[1-3], however, Arabic script does not lend itself easily to the automatic recognition based on today's technology.

Arabic script consists of 28 basic characters; most of them have different shapes for different instances. Difficulties in Arabic character scripts are due to its expressive richness as a language. The shapes are slightly complicated and context-sensitive too; character shapes changing with changes in place, the preceding character or the succeeding one. At times even the 3rd, 4th or 5th character may cause a similar change as depicted in an n-gram model in a Markov chain. The use of single, double or triple dots in various position for many characters (initial, medial, final or isolated), the centering of the letters in the text and most difficult of all is the so many vowel signs attached to most characters (if implemented). Besides, words written from right to left (RTL)[4]. For these reasons, Arabic script is considered to be a difficult one with a much richer character set than Latin having features that make direct application of algorithms for character classification in other languages difficult to achieve[5]. A comprehensive summary of the Arabic script characteristics and features is given by Perviz et al.[6].

The Latin, Chinese and Japanese scripts which have received ample research and work has been done on the optical recognition of these scripts. Compared to this, only few studies have specifically addressed the recognition of Arabic text. This is due to the complexity of the Arabic script itself while a lack of interest in this regard accounts for another[7]. Khorsheed *et al.*[8] presented an approach in which the system recognizes an Arabic word as a single unit using a Hidden Markov Model. The system depends greatly on a predefined lexicon, which acts as a look-up dictionary. All the segments in a word are extracted from its skeleton, and each of the segments is transformed into a feature vector. Then each of the feature vectors is mapped to the closest symbol in the codebook. The resulting sequence of observations is presented to a Hidden Markov Model for recognition. Fanton[9] has discussed the features that Arabic writing and identified the fact that these features impose computational overload for any Arabic software. He also noted that the way in

Corresponding author: Hamza A.Ali, Faculty of Computer Science and Information Technology, Al- Isra Private University, P.O.Box 22, Code 11622, Amman, Jordan

which Arabic is printed imitates handwriting. He pointed out that Finite State Automata give an efficient solution for the translated problems, which can be formalized as regular languages.

Chen *et al.*[10] addressed the problem of automatic recognition of an image pattern without any consideration of its size, position and orientation. In this regard, the extracted image features are made to have properties that are invariant with image transformation including scale, translation and rotation. They approximated the transformation by affine transformation to preserve co linearity and ratios of distances.

Optical Character Recognition (OCR) is one of the most successful applications that have been proposed for Artificial Neural Networks (ANN's)[11]. They lend themselves to be highly applicable for OCR as compared with statistical, syntactical or structural approaches[12]. NN's have faster development times, they have an ability to automatically take into account the peculiarities of different writing/printing styles, and they can be run on parallel processors. On the other hand, introducing a new shape to the NN requires that the network be retrained or even worse, that the network be trained to a different architecture.

ANN is a non-linear system which may be characterized according to a particular network topology[13]. This topology is decided by the characteristics of the neurons and the learning techniques. OCR utilizes the main advantages of ANNs, i.e., their fast development times, ability to automatically take into account the peculiarities of different writing/printing styles, inherent ability of parallel processing, retraining for new shapes or different architecture. An intensive work can be found in the subject of Arabic OCR using neural networks[14]. ANNs can simply cluster the feature vectors in the feature space[15] or they can integrate feature extraction and classification stages by classifying characters directly from images[16]. NNs were also applied for recognition of Arabic words on-line[17].

ANN's are simply processing structures having many simple, highly connected processing elements that can process information by its dynamic state response to external inputs, which means that they have certain characteristics with great similarities to those of biological neural systems.

Generally speaking, the common architecture of ANNs used in Arabic OCR is a network with three layers: input, hidden and output, as illustrated in Fig. 1. The number of nodes in the input layer varies according to the dimensionality of the feature vector or the segment image size. The number of nodes in the hidden layer governs the variance of samples that can be correctly recognized by this ANN[18] and the number of nodes in the output layer corresponds to the number of samples to be recognized.

Each node represents an ANN element may have many input signals but it is limited to one output signal. It may have a set of continuous or discrete inputs, connected through links from previous neurons, x's. Each link has an adaptive coefficient called synaptic weight, w assigned to it.

ANN's can be classified into forward propagation and back propagation networks. Forward propagation (or feed-forward) networks are called "Non-Recurrent" and they have no feedback connections that connect through weights expended from the output layer to the inputs of the same or previous layers, while back propagation (or feed-back) networks are called "Recurrent" and they contain feedback connections. Recurrent networks recalculate previous outputs back to inputs hence; output is determined both by their current input and their previous outputs. For this reason, recurrent network can be regarded very similar to short-term memory in humans in that the state of the network outputs depends: upon their previous input. The Hopfield model is the simplest and most widely used feedback neural architecture. Another example of feedback network is Boltzman machine, which is close to Hopfield model architecture.

The learning ability of ANN's is the basic feature of intelligence. It implies that the processing element somehow changes its input/output behavior in response to the environment. In a similar manner to the way that a child learn to identify various things, ANN learns by example[19], i.e., by repeatedly trying to match that set of input data to the corresponding required output. Therefore, after a sufficient number of learning iterations, the network modifies the weights in order to obtain the desirable behavior pattern for new input conditions.

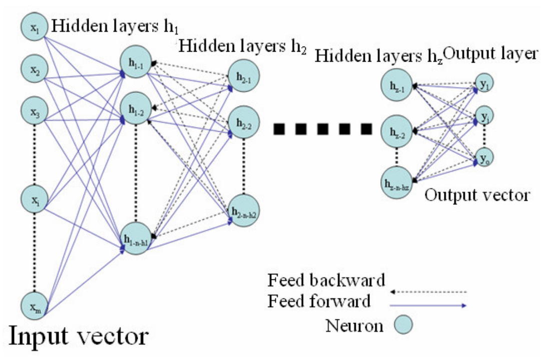


Fig 1: General structure of back propagation layered neural network

Arabic Character Set Review: Arabic language is a highly developed language with basic letter set consists of 28 characters, they are

ا، ب، ت، ث، ج، ح، خ، د، ذ، ر، ز، س، ش، ص، ض، ط، ظ، ع، غ، ف، ق، ك، ل، م، ن، ه، و، ي

Arabic language can be written in so many different font shapes, for example Simplified Arabic, Kufi, Andalusi and Hejaz. Arabic characters are used in writing many languages not only in Arabic countries, but for Urdu and Farsi and other languages in countries where Islam is the principal religion (such as Iran, Pakistan and Malaysia). The special characteristics of Arabic written words and characters do not allow the direct application of algorithms of other languages. They have so many interesting but complicated features. A summary of the most important features may contain the following:

- Cursive; it is cursive and written from left to right.
- Multiple shape; each character has 2-4 shapes depending on its position within the word, initial, median or final.
- It might be connected or not connected to the previous or/and the next character in the word, for example, the letter ب, may come at the beginning برد, in between and connected مبرد, at the end and connected طلب and at the end and not connected مطلوب.
- Dots; Many letters of the Arabic alphabet have dot (or dots), above or below the character body, such as ب، ي، ن، ت، ث، ج، ف.
- Hamza; some letters may have a "Hamza" (zigzag shape) on top or below its body, such as ؤ, أ, إ ـئ and ئ.
- Madda; some letters have "Madda", to form a new character, such as letter (ا) to become (آ), e.g. آحاد.
- Vowels; Arabic language contain many vowel sub-characters, (i.e., ُ ٌ َ ً ِ ٍ). Any letter may take one of many vowels.
- Su-koon; a sign for cooling down the vowel, called su-koon (ْ), that sits on the letters resulting in no vowel effect.
- Overlapping; some Arabic's characters become over each other horizontally when they connected with each other.

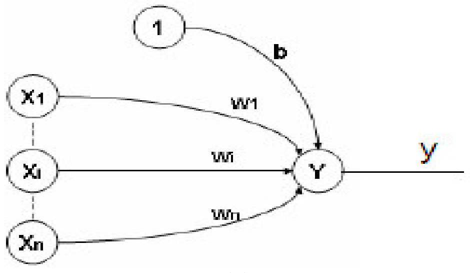
In addition to the letters and their vowels, the character set includes 10 numerals (i.e., 0 - 9) and a long list of special characters, (e.g. * / + - [ ] !).

When all characters, vowels, special symbols and numerals are included, the combinations of expected characters may go over one thousand shapes, which makes recognition or classification task an extremely complicated problem. However, one may starts with the basic characters and numerals only, i.e., 38 characters, as we did in this study. The objectives of this study were to: (i) summarize the main characteristics of Arabic language writing style. (ii) suggest a neural network recognition circuit.

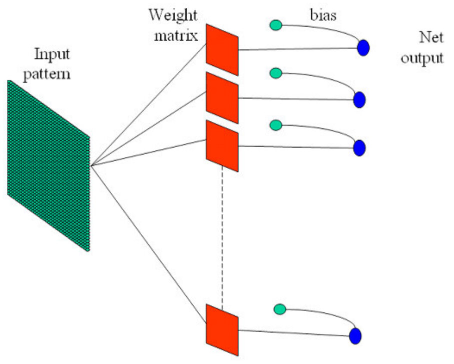
MATERIALS AND METHODS

The design, training and testing of two networks for recognition of hand written Arabic alphanumeric characters namely Natsha net and BP-NN net are described below. The training and testing is achieved with the aid of Microsoft office.

Natsha net (perceptron-like net)[20]: This neural network simply combines the perceptron principle for pattern classification together with the ADALINE circuit in single layer NN that consists of input and output layers with a bias, as illustrated in Fig. 2.



(a)



(b)

Fig 2: Architecture of perceptron-like module. (a) Neural network architecture, (b) Computation representation

The input layer (corresponds to the retina in the visual system) consists of 50x50 pixels matrix. This size is found necessary to accommodate the input Arabic characters. While the output layer consists of as many neurons as the required number of characters to be recognized. Each output neurons has a bias and connected via weight matrix to the output layer. Bipolar sigmoid (i.e., +1 or -1) activations are used for both input and target signals. This network is a supervised net that is a feed forward only.
The total input to the jth neuron output neuron is calculated by:

$$y_{i-in} = \sum_{i=1}^{n} w_{ij} X_i + b_i \qquad (1)$$

$$y_{i-in} = \sum W_{ij} X_i + b_i \; i=1$$

Where, b_j is bias weight for the jth neuron.

The weights are given initial zero values, then adjusted by adding the weight difference Δw_{ij} and Δb_j to the previous values after each epoch. Where:

$$W_{ij\,new} = W_{ij\,old} + \alpha(t_j - y_{j-in})X_i \qquad (2)$$

$$b_{j\,new} = b_{j\,old} + \alpha(t_j - y_{j-in})$$

Where:

α	= Learning rate
t_j and y_{j-in}	= Target and actual values for the jth output neuron, respectively.

The values for x_{ij}, y_{j-in} and t_j are either +1 or -1.

The activation output for the jth neuron Y_j is calculated, using bipolar sigmoid function as:

$$Y_j = f(y_{j-in}) = -1 \text{ if } y_{j-in} < \theta \qquad (3)$$

$$= 0 \text{ elsewhere}$$

Where, θ = threshold value. It is found that $\theta = 2$ and $\alpha = 0.1$ were suitable choices.

Multilayer BP-NN: A multilayer neural network using back propagation training policy that consists of an input layer, one hidden layer and an output layer is designed as shown in Fig. 3. The input layer consists of 50x50 pixels, hidden layer 20x20 neurons while the output layer consists of as many neurons as the required number of characters to be recognized. Each neuron in the hidden layer and the output layer has a bias and connected via weight matrix to the previous layer.

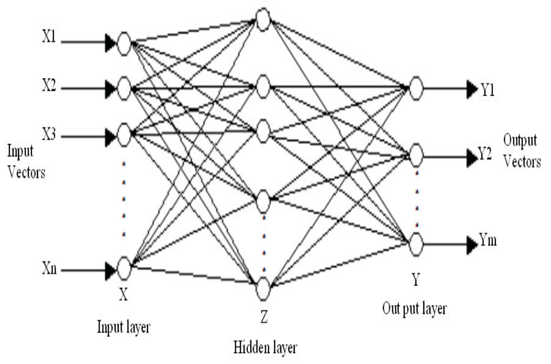
Using training data (input-target pairs), the weights of the neuron can be iteratively adjusted to give local or global optima. Optimum weights in the sense of least square errors were derived by Widrow and Hoff [21]. It is called the LMS algorithm and is commonly known as the Widrow-Hoff rule. In the LMS algorithm, the network weights are moved along the negative of the gradient of the performance function. Specifically, after each iteration or epoch the weights to the output layer are adjusted using a delta signal that in terns is used to adjust input connection weights to these neurons by multiplying error by the derivative of the sigmoid function evaluated as the net output value for that neuron which can be expressed mathematically as:

$$\delta_k = (t_k - y_k).f'(y_{k-in}) \qquad (4)$$

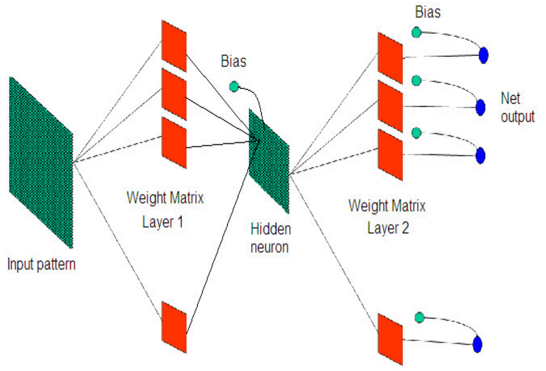
Where, t_k target output for the kth output neuron. These values of deltas are used to adjust their input connection weights according to the following formula:

$$W_{ik\,new} = w_{ik\,old} + \alpha.\delta_k.Z_j \qquad (5)$$

α is the learning rate, typically in the range of 0 to 1.



(a)



(b)

Fig. 3: The Multilayer BP-NN. (a) Neural Network architecture, (b) Computation representation

The difficulty arises when training internal layer weights as no targets is available. The solution lies in propagating the error back, layer by layer, from the output successively to backwards layers. The j^{th} hidden neuron receives an error signal as the weighted sum of the following layer error signals, then multiplies this summation by the derivative of the input to the neuron, i.e.:

$$\delta j = f'(Zk - in)\sum_{k=1}^{n} \delta k . Wik \qquad (6)$$

This signal is used to adjust the input connection weight V_{ij} to the hidden neurons Z_j, i.e.:

$$V_{ij\,new} = V_{ij\,old} + \alpha . \delta_i \qquad (7)$$

Thus neuron uses its error signal to train its associated weights, and then passes it back to all neurons to which it is connected in the previous layer. This two-step process is repeated over the training set elements until the network converges and produces the desired response. It must be noted that all weights were set to zero initially.

The Learning Tool: The BP-NN network was trained and tested using a software system designed to work according to the plan outlined in Fig. 4. It is a generic design to be implemented for the use of any character set combination available in MS office. The implementation process of this tool can be summarized as follows.

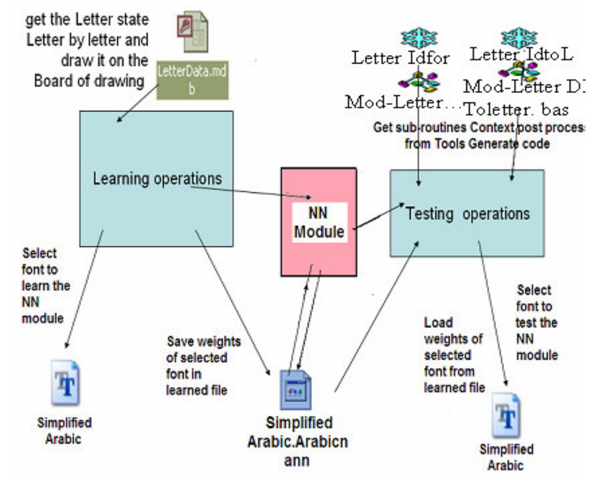

get the Letter state Letter by letter and draw it on the Board of drawing
LetterData.mdb
Letter Idfor
Letter IdtoL
Mod-Letter...
Mod-Letter D
Toletter. bas
Get sub-routines Context post proces from Tools Generate code
Learning operations
NN Module
Testing operations
Select font to learn the NN module
Save weights of selected font in learned file
Load weights of selected font from learned file
Select font to test the NN module
Simplified Arabic
Simplified Arabic.Arabicn ann
Simplified Arabic

Fig. 4: Processes plan for the proposed networks

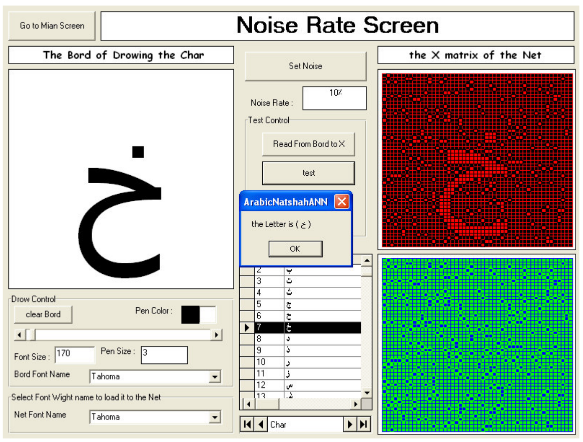
To train the neural network, first a character set font is selected together with the font size. It is interred to the database by the letter state tool. Its characters are then taken one by one to a drawing board that transfers them to the input matrix. The weights Matrix is iteratively computed and is then saved into a file called "Simplified Arabic.Arabic.ann", to be then used at the testing process.

RESULTS

Implementation: Implementing the selected module, i.e., running either the perceprton-like module or the back propagation module described above means performing the required processes including five functions, which are; learn net, view weights, test net and find test rates. When learn net is selected, it gives a screen that allows for selecting letter font, letter size, start learn and stop learn. All character sets included in Microsoft word are available to be used for network training, and then calculated weights are saved in a database. These weights can be viewed, and then finally you can enter any character to test for recognition.

Noise is also added to any character in order to see its effect on the recognition for the selected character. Figure 5 shows an example of one screen that illustrates how noise at a certain rate can be added to the input and recognition performance is checked and reported. This screen includes selecting the character set, its font type, size and the noise percentage required to be added to each input character.

Moreover, the programming tool used shows other computation information, such as testing recognition speed, number of epochs, number of patterns correctly recognized, number of patterns wrongly recognized and number of rejected patterns.



the Letter is (خ)

Noise Rate : 10%

Font Size : 170 Pen Size : 3

Bord Font Name Tahoma

Select Font Wight name to load it to the Net

Net Font Name Tahoma

2	ب
3	ت
4	ث
5	ج
6	ح
7	خ
8	د
9	ذ
10	ر
11	ز
12	س
13	ش

Fig. 5: Test screen for character recognition with noise rate selection

Table 1: Training and testing times for the network

		Measured time			
		Perception net		BP-NN	
Font type	Character set	Training	Ave. testing	Training	Ave. testing
Arial	38-arabic	824	0.041	1530	0.075
(Arabic)	52-latin	1251	0.043	2102	0.060
	10 Numerals	30	0.033	245	0.045
Andulus	38 Arabic	959	0.040	711	0.212
	10 Numerals	26	0.037	157	0.055
Simplified	38 Arabic	768	0.044	1357	0.077
Arabic	52 Latin	1615	0.041	2433	0.075
	10 Numerals	25	0.036	166	0.049
Tahoma	38 Arabic	1419	0.041	2433	0.075
	52 Latin	335	0.042	870	0.060
	10 Numerals	25	0.033	390	0.040

Table 2: Testing rates with no addition of noise

		Recognition rate %			
		Perception net		BP-NN	
Font type	Character set	Correct	Error	Correct	Error
Arial	38-arabic	100	0	100	0
(Arabic)	52 Latin	96	4	97	3
	10 numerals	100	0	100	0
Andulus	38 Arabic	100	0	100	0
	52-latin	100	0	100	0
	10-numerals	100	0	100	0
Simplified	38 Arabic	100	0	100	0
	52 Latin	96	4	98	2
	10 numerals	100	0	100	0
Tohama	38 Arabic	100	0	100	0
	52 Latin	100	0	100	0
	10 numerals	100	0	100	0

Table 3: Testing rates with addition of 10% noise

		Recognition rate %			
		Perception net		BP-NN	
Font type	Character set	Correct	Error	Correct	Error
Arial	38-arabic	79	21	89	11
(Arabic)	52 Latin	81	19	94	6
	10 numerals	100	0	100	0
Andulus	38 Arabic	79	24	85	15
	52-Latin	88	10*	92	8
	10-numerals	100	0	100	0
Simplified	38 Arabic	82	18	93	7
	52 Latin	87	13	95	5
	10 numerals	100	0	100	0
Tohama	38 Arabic	79	18*	91	9
	52 Latin	83	17	94	6
	10 numerals	100	0	100	0

*: Note: There a rejected recognition is found in this case

The programs written to implement the proposed networks are designed to be trained for any character set type, however only four different types of Arabic character are involved in the experiment of this study, namely Arial, Andalus, Simplified Arabic and Tuhama fonts.

Table 4: Testing rates with addition of 20% noise

		Recognition rate %			
		Perception net		BP-NN	
Font type	Character set	Correct	Error	Correct	Error
Arial	38-arabic	53	47	89	11
(Arabic)	52 Latin	62	38	93	7
	10 numerals	80	20	97	3
Andulus	38 Arabic	55	45	88	12
	52-Latin	77	23	90	10
	10-numerals	100	0	100	0
Simplified	38 Arabic	55	42*	86	14
	52 Latin	76	33	92	8
	10 numerals	100	0	100	0
Tohama	38 Arabic	79	21	95	5
	52 Latin	62	38	89	11
	10 numerals	100	0	100	0

*: Note: There a rejected recognition is found in this case

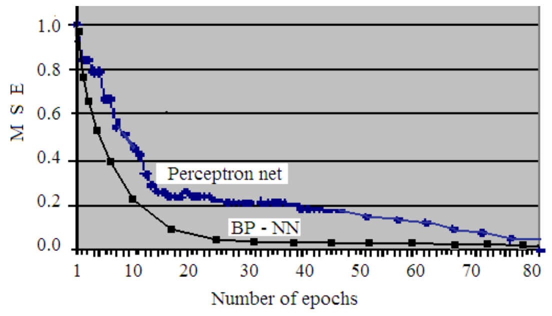


Fig. 6: Mean Square Error measurements for the basic Arabic (Arial) Character set recognition

Besides, three character combinations only were investigated, namely the basic Arabic alphanumeric set (28 letters + 10 numerals), the Latin alphabetic set (52 lower case and upper case) and the 10 Arabic numerals on their own.

The time taken to train the networks for the character sets as well as the time required to test the net for the recognition of all the elements of the corresponding sets are calculated and listed (Table 1). These measurements are for the four chosen fonts for both the perceptron-like net and the BP-NN.

Observed training and testing times for the proposed perceptron and BP-NN networks varies for various fonts. This may be attributed to the different features found in each set font type. Simplified Arabic type is found to be the fastest to train among the four sets under study, while testing time has no considerable differences. These results show that training the numerals set only is the fastest. This speed can be attributed to their small number as compared with other sets. The BP-NN network needed more time for both training and testing due to its size and complexity as compared with the perceptron, however this time consumption can be considered as a price for their improved recognition accuracy.

In order to see how the network recognition improves with increasing number of iterations, the mean square error MSE is calculated and plotted in Fig. 6. It is measured for repeated training with increasing number of epochs for both perceptron-like net and the BP-NN, however it is only presented for the 38 Arabic (Arial) alphanumeric set.

Recognition rates for both perceptron-like and BP-NN are computed for the four types of Arabic character sets under consideration without introducing any noise (Table 2). Then different levels of noise were added to the character images in the range up to 20% for the cases understudy and recognition efficiency in the form of correct recognition and error recognition rates are calculated and tabulated (Tables 2-4).

DISCUSSION

For no noise calculations (Table 2), full recognition was obtained for all Arabic fonts, while 96% correct recognition is obtained for the Latin alphabetic set and 4% wrong recognition. Moreover, no rejection rate was noticed.

The addition of 10% noise level to the image is investigated and listed (Table 3). This has resulted into an increased recognition error of up to 24% for Arabic (Andalus) alphanumeric set. However, it still gives full recognition for the numerals. Few rejections are also noticed as in the case of Andulus and Simplified Arabic fonts when the Latin set is chosen.

When the noise level increased to 20%, more error recognition is observed (Table 4). Correct recognition calculations for both Arabic and Latin letters show drastic deterioration, but those for numerals look unaffected for most fonts. It is noticed that some error for Arabic (Arial) numerals set only still exist. However, no rejection was noticed when BP-NN is implemented for all tested font types.

The recognition error or the mean square error decreases exponentially with the increase in number of training epochs (Fig. 6). It is found that the error is extremely reduced after about 80 epochs, which can be considered as good enough for a practical system. It is found too that the MSE reduces for the BP-NN faster and stabilizes at lower rate as compared with the perceptron net.

CONCLUSION

This study proposes feed forward neural network that combines properties of perceptron and ADALINE net as well as multilayer neural net with back propagation training policy. They are trained and tested for implementation of Arabic script classification. It also included the design of a software tool suitable for the training and testing NN for any character set combinations, sizes or fonts utilizing MS word. The effect of the presence of noise in the original script is also investigated. The scheme takes the whole character as one feature. The results have shown considerable degree of confidence in recognition printed Arabic characters with reasonable degree of reliability in a noisy environment. It was evident that as the size of the character sets increases, the required time for training is increased considerably and the recognition rate decreases especially when noise rate is increased.

ACKNOWLEDGMENT

The researcher acknowledges the academic support of Isra Private University. Also thanks are due to Mr. Rajab Natsha for his assistance in programming efforts.

REFERENCES

1. Asiri, A. and M.S. Khorsheed, 2005. Automatic processing of handwritten Arabic forms using neural networks. Proceeding of the World Academy of Science, Engineering and Technology, Aug. 2005, pp: 313-317. http://www.waset.org/pwaset/v7/v7-61.pdf.
2. Sarhan A.M. and O. Helalat, 2007. Arabic character recognition using ann networks and statistical analysis. Proceeding of European and Mediterranean Conference on Information Systems, June 24-26, Polytechnic University of Valencia, pp: 1-9. http://www.iseing.org/emcis/EMCIS2007/emcis07cd/EMCIS07-PDFs/673.pdf.
3. Alnsour, A.J. and L.M. Alzoubady, 2006. Arabic handwritten characters recognized by neocognitron artificial neural network. J. Pure Appl. Sci., 3: 1-17. https://www.sharjah.ac.ae/English/About_UOS/UOSPublications/Appliedsciences/Issues/Documents/3_2/AlNsour_1_e.pdf.
4. Hadjar, K. and R. Ingold, 2003. Arabic newspaper segmentation. Proceeding of 7th International Conference on Document Analysis and Recognition, Aug. 3-6, IEEE Computer Society, pp: 895-899. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1227789.

5. Altuwaijri, M.M. and M. Bayoumi, 1994. Arabic text recognition using neural networks. International Symposium on Circuits and Systems, May 30-June 2, IEEE Xplore, London, UK ., pp: 415-418.
DOI: 10.1109/ISCAS.1994.409614.
6. Ahmed, P. and Y. Al-Ohali, 2000. Arabic character recognition: Progress and challenges. J. King Saud University, Comput. Inform. Sci., 12: 85-121. http://digital.library.ksu.edu.sa/V12M152R682.pdf.
7. Syed, M. S., N. Nawaz and A. Al-Khuraidly, 2003. Offline arabic text recognition system. International Conference on Geometric Modeling and Graphics, July 16-18 IEEE Xplore, London, UK., pp: 30-35. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1219662.
8. Khorsheed M.S. and W.F. Clocksin, 1999. Structural Features of Cursive Arabic Script. Proceeding of the 10th British Machine Vision Conference, Sept. 13-16, Nottingham, UK., pp: 422-431.
http://www.bmva.ac.uk/bmvc/1999/papers/42.pdf
9. Fanton, M., 1998. Finite state automata and arabic writing. Workshop on Computational Approaches to Semitic Languages, Aug. 16-16, University of Montreal, Canada, pp: 26-33. http://www.aclweb.org/anthology-new/W/W98/W98-1004.pdf
10. Chen, Q., E. Petriu and X. Yang, 2004. A comparative study of fourier descriptors and hu's seven moment invariants for image recognition. IEEE Canadian Conference on Electrical and Computer Engineering, May 2-5, Niagara Falls, Canada, pp: 103-106. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1344967.
11. Khorsheed, M.S., 2000. Off-line Arabic character recognition-a review. Patt. Anal. Appl., 5: 31-45. DOI: 10.1007/s100440200004.
12. Lucas, S., E. Vidal, A. Amiri, S. Hanlon and J.C. Amengual, 1994. A comparison of syntactic and statistical techniques for off-line OCR. Lecture Note Comput. Sci., 862: 168-179. DOI: 10.1007/3-540-58473-0_146.
13. Pao, Y., 1989. Adaptive Pattern Recognition and Neural Networks. Addison Wesley Pub., Company, Inc., ISBN-10: 0201125846.
14. Hassibi, K., 1994. Machine-printed Arabic OCR using neural networks. Proceeding of the 4th International Conference and Exhibition on Multi-Lingual Computing, April 1994, Cambridge University Press, Cambridge, UK., pp: 2.3.1-2.3.12.
15. Mowlaei, A., K. Fayez and A. Haghighat, 2002. Feature extraction with wavelets transform for recognition of isolated handwritten Farsi/Arabic characters and numerals. Proceeding of the 14th International Conference on Digital Signal Processing, pp: 923-926. DOI: 10.1109/ICDSP.2002.1028240.
16. Mosfeq, R., 1996. Arabic character recognition using integrated segmentation and recognition. The 5th International Conference and Exhibition on Multi-Lingual Computing, April 1996, Cambridge University Press, Cambridge, UK., pp: 2.1.1-2.1.10.
17. Abdelazim, H. Y., 1996. A Hybrid Fuzzy-Neural Approach to the Recognition of Arabic Script, Proc. of 5th Int. Conf. and Exhibition on Multi-Lingual Computing, April 1994, Cambridge University Press, Cambridge, UK., pp: 2.3.1-2.3.13.
18. Said, F., R. Yacoub and C. Suen, 1999. Recognition of English and Arabic numerals sings a dynamic number of hidden neurons. Proceeding of the 5th International Conference on Document Analysis and Recognition, Sept. 20-22, IEEE Computer Society, Washington, DC,, USA,, pp: 237-241.
19. Ali, H.A. and M.A.B. Mahmood, 2002. A hybrid neural network model for character recognition of hand written numerals. J. Associ. Advance. Model. Simulat. Tech. Enterprises45: 53-67.
20. Ali, H.A. and R.M. Natsha, 2003. Neural networks tool for arabic script classification. Proceeding of the Arab Conference on Information Technology, Dec. pp: 285-297.
21. Canny, J.F., 1986. A computational approach to edge detection. IEEE Trans. Patt. Anal. Mach. Intell., 8: 679-698. http://portal.acm.org/citation.cfm?id=11274.11275.