Neural Networks Tool for Arabic Script Classification

Hamza A. Ali. and Rajab M.Z.R.Natshah

Zarka Private University , Jordan

alsewadi@hotmail.com and rajab_n@yahoo.com

Abstract:

A neural network that combines some properties of perceptron net with ADALINE net is developed for classification of Arabic script. It is based on supervised learning or with tutor technique. A software tool is designed for training and testing any set of character combinations or fonts. The circuit is tested for various combination sets of Arabic characters with different fonts and sizes. The network exhibited high recognition and low error rates having reasonable tolerance to some noise level.

Key words

Artificial Neural Networks, Neocognitron, Pattern recognition, Character Recognition, Arabic Script Classification.

1. Introduction:

Recognition of typed Latin characters has long been implemented [1,2,3]. However, research work on Arabic character recognition has still long way to go. Arabic script does not lend itself easily to the automatic recognition based on today's technology.

Difficulties in Arabic character scripts are due to its expressive richness as a highly developed language. Technically, it may be attributed to many reasons of which, the slightly complicated shape of individual characters, the change of shape for the same character depending on its position in the word, the use of single, double or triple dots in various position for many characters, the centering of the letters in the text, the so many different letters concatenation rules and most difficult of all is the so many vowel signs (when included) attached to most characters.

Much interest in the use of neural network has grown tremendously for character recognition of Arabic script. Artificial Neural Networks (ANN's) lend themselves to be highly applicable for character recognition as compared with statistical, syntactical or structural approaches [3]. 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 [4,5]. This means that they have certain characteristics with great similarities to those of biological neural systems. Each neural element or the building block of NN's may have many input signals but it is limited to one output signal as illustrated in figure 1 [6]. It has a set of continuous or discrete inputs, connected

through links from previous neurons, x's. Each link has an adaptive coefficient called synaptic weight, $\boldsymbol{w}$ assigned to it.

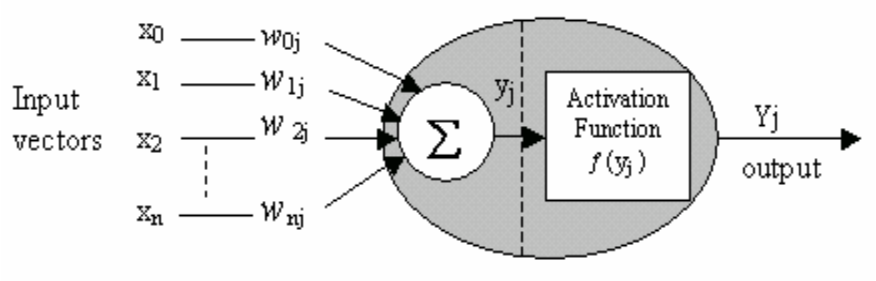

Fig 1. Processing functions inside the neuron element.

Each synaptic weight might result into amplification, attenuation or possibly changing the sign of the signal in the link. The output of the j^{th} neuron can be calculated by equation (1).

(1)

Where, $\boldsymbol{w}_{ij}$ is the weight of the i^{th} input vector to the j^{th} neuron. Furthermore, y_j is processed by an activation function $f(\ )$ in order to give the final neuron's output activation signal as in equation (2).

$$Y_j = f(y_j) \qquad (2)$$

The activation function determines the processing inside the neuron. It might be linear or non-linear function depending on the proposed network. However, the function limits the output of the neuron to the desired numerical range. Typically, limiting the output between 0 and 1 or between –1 and +1 for binary or bipolar sigmoid function, respectively.

Numerous numbers of such processing elements form an Artificial Neural Network, ANN as shown in figure 2. In layered neural network system, the input signals come from the outside world. This relationship between input vectors and output signals are determined usually by first order ordinary differential equations. Auto-adjustment to the coefficients of the differential equations gives the neural networks its ability to adjust the values of its internal variables. This ability of self-adapting dynamic system that can notify its own response to external forces is the outstanding feature of the NN's. The basic phenomena of neural learning, is that each link has its own small local memory, which stores the result of some previous computation as an adaptive coefficient.

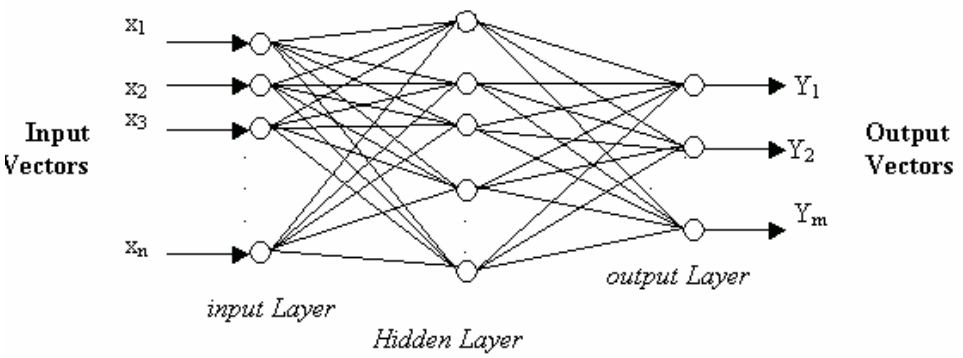

Fig 2. General structure of layered Neural Network

The learning ability of these networks is the basic feature of intelligence [7, 8, 9]. 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, NN learns by example [10, 11, 12, 13], 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 neural network modifies the weights in order to obtain the desirable behavior pattern for new input conditions [14]. ANN's can be classified by their learning scheme as supervised, un-supervised or batch learning which correspond to learning with a tutor, no tutor or encoding, respectively.

After this brief introduction in section 1, a short Arabic script review is presented in section 2 together with description of a program organization for Arabic script preparation. The proposed network for character classification is outlined in section 3. Section 4 lists out some of the results obtained and finally section 5 concludes the work.

2. Arabic Script review

Arabic language 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)
In addition to the so many different fonts, such as Simplified Arabic, Kufi, Andalusi etc., they have so many interesting but complicated features. A summary of the most important features (but not the least) may contain the following:

(1) It is written from right to leftt.
(2) Any of the basic letters may take different shapes depending on its position in the word whether it comes as the first character, in between character or at the end of a word.
(3) It might be connected or not connected to the previous and / or the next character in the word, for example, the letter **ب** *may come at the beginning e.g.* **بسم** *in between and connected e.g.* **معبد**, *at the end and connected e.g.* **نذهب** *and at the end and not connected e.g.* **مغرب**
(4) An additional special character called hemza (ء) may come alone or join many letters

to form a new character, such as ؤ إ أ ـئـ *and* ئ.

(5) A connecting sign comes on the first letter (ا), like (آ), e.g. آحاد.

(6) A sign for cooling down the vowel, called su-koon,) ْ) , that sits on the letters resulting in no vowel effect.

(7) Any letter may take any of the vowels, (they are ُ ٌ ّ َ ً ِ ٍ ْ*).*

In addition to the Arabic letters and their vowels, the character set would include 10 numerals (i.e. 0 – 9) and a long list of special characters, (such as * / + - [ ] …).

A study of how to generate the correct shape for any letter is conducted and a program is written to facilitate a tool that stores each Arabic character as one ASCII code, but can decide its shape according to its position in the words. Figure 3 shows the ER diagram for the letter state algorithm of Arabic script. The letters and their states are store in a database to be called when needed in the learning program.

When all characters, letters with vowels, special symbols and numerals are included, the combinations of expected characters may go over one thousand shapes, which makes recognition or classification task a very complicated problem. However, one may starts experimenting with the basic characters and numerals only, i.e. 38 characters, as we did in this work.

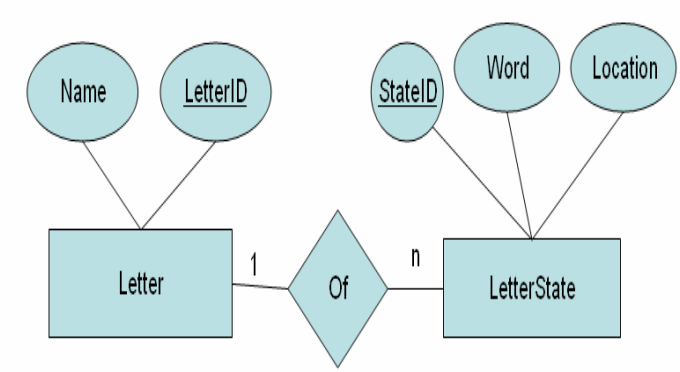

Fig 3. ER diagram of the letter state algorithm

3. The Proposed Network

3.1 Network Architecture

The proposed neural network is a try to simply combine the perceptron principle for pattern classification together with the Adaptive Linear Neuron (ADALINE) circuit. The bipolar sigmoid, (i.e. +1 or –1) activations for both input and target signals is adopted. The suggested network consists of two layers only, an input layer and output layer with a bias, as shown in figure 4.

The input layer in this network corresponding to the retina in the visual system consists of 50x50 pixels matrix. This size is found necessary to accommodate the input Arabic characters for an acceptable recognition results. While the output layer consists of as many single neurons as the required number of characters to be recognized, i.e. each output neurons corresponds to one character. Each output neuron has a bias and

connected via weight matrix to the input layer. The network assembly for one output neuron may be considered as one network element, we called it Natshah net element. This network is a supervised net that is a feed forward only.

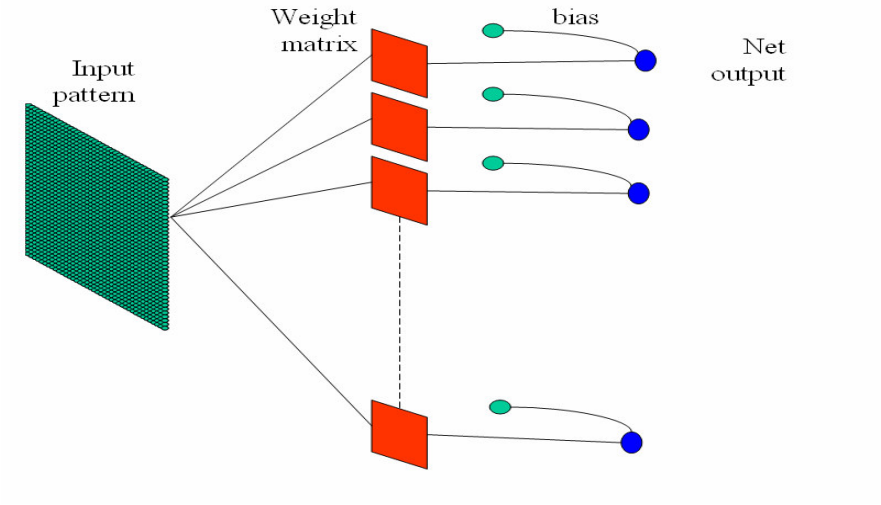

Fig 4. The architecture of neural network

The total input to the jth output neuron is calculated by

(3)

where b_j is the bias synaptic weight for the j^{th} neuron.

The weights are given initial values of 0's, then they are adjusted by adding the weight differences Δw_{ij} and Δb_j to the previous values after each epoch, given by.

$$\Delta w_{ij} = \alpha (t_j - y_j) x_i$$
$$\Delta b_j = \alpha (t_j - y_j) \qquad (4)$$

where α is the learning rate, t_j and y_j are target and actual values for the j^{th} output neuron, respectively. The values for x_{ij}, y_j and t_j are either +1 or –1.

The activation output for the j^{th} neuron Y_j is calculated, using bipolar sigmoidal function as

$$Yj = f(yj) = \begin{cases} +1 & \text{if } y_j > \theta \\ -1 & \text{if } y_j < \theta \\ 0 & \text{elsewhere} \end{cases} \qquad (5)$$

where θ is a threshold value.

In our computations, we found $\theta = 2$ and $\alpha = 0.1$ are suitable choices. The initial weights are taken as zeros and the stopping criteria for ending the search is for the summation of weight adjusting values becomes zero or negligible.

3.2 The Natshah Net Learning Tool

A software program is developed in Visual Basic programming to provide a tool for the network training and testing according to a plan outlined in figure 5. It is designed generally for implementing the network in training and testing the network using any character set combination. The main idea of this tool can be summarized as follows.

To learn a number of vectors (characters), they are inserted in the input to the database by the letter state tool. Then they are taken one by one to a drawing board that transfers them to the input matrix X. At the end of the training process, the computed weights are saved into a file to be then used at the testing process.

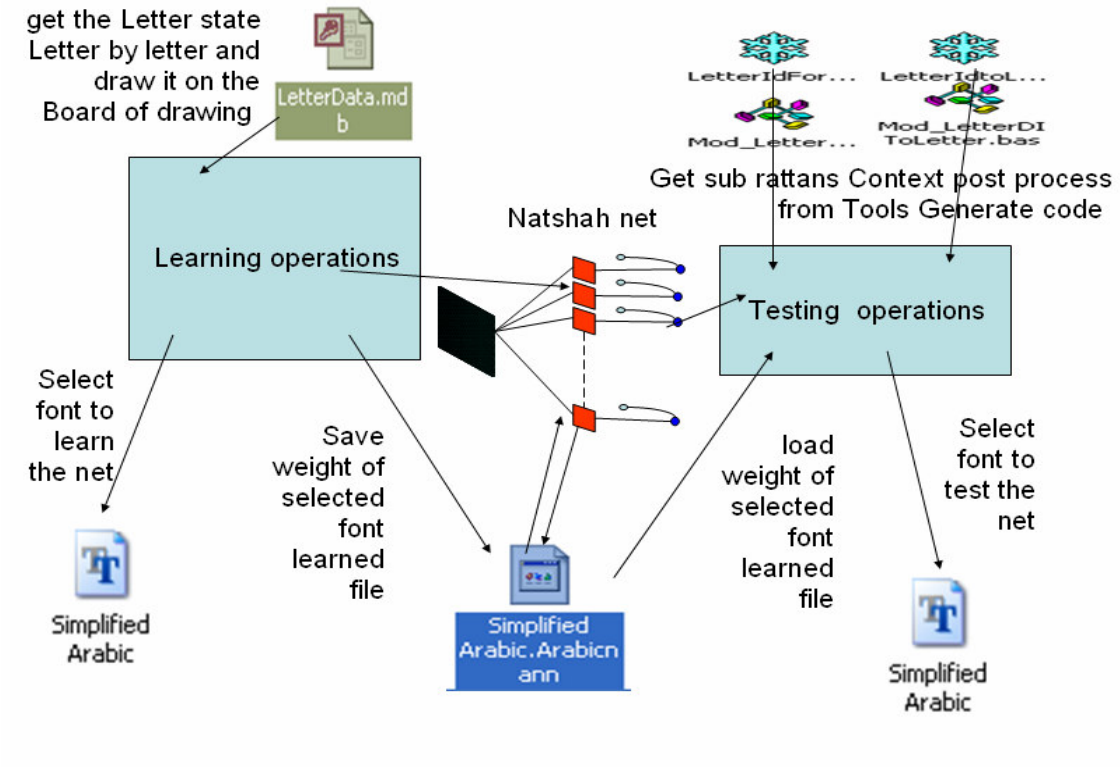

Fig 5. The operation plan for the learning tool.

Running the tool gives the main output screen that has switches to four functions besides exit, i.e. learn net, view weights, test net and find test rates, see figure 6.

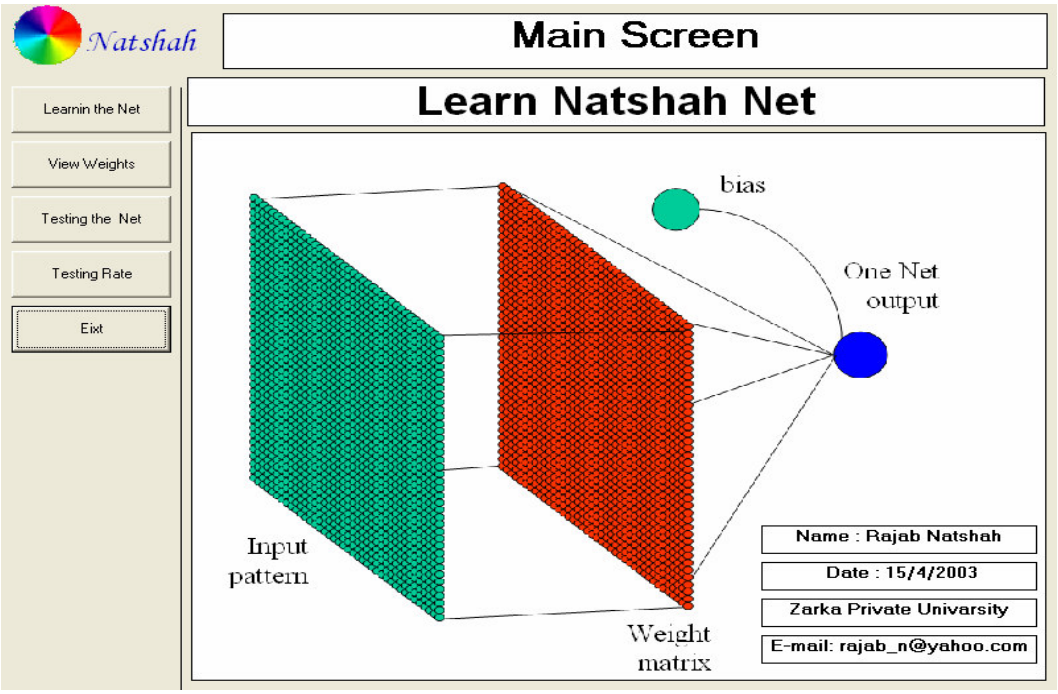

Fig 6. The main screen for the network tool.

Selecting learning net, gives a screen that allows for selecting letter font, size, start learn and stop learn. Then the selected set of characters is used for the network training and the weights are saved. These weights can be viewed, then finally you can enter any character to test for recognition as shown in figure 7.

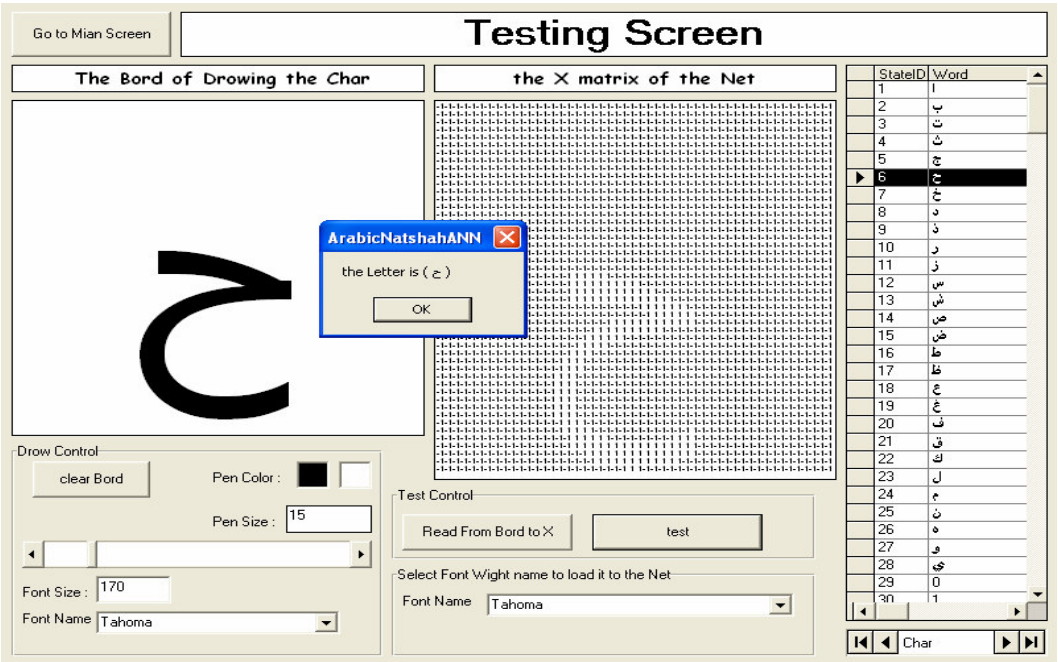

Fig 7. The testing of characters screen.

Noise can also be added to any character in order to see their effect on the recognition for the selected character, as shown in figure 8.

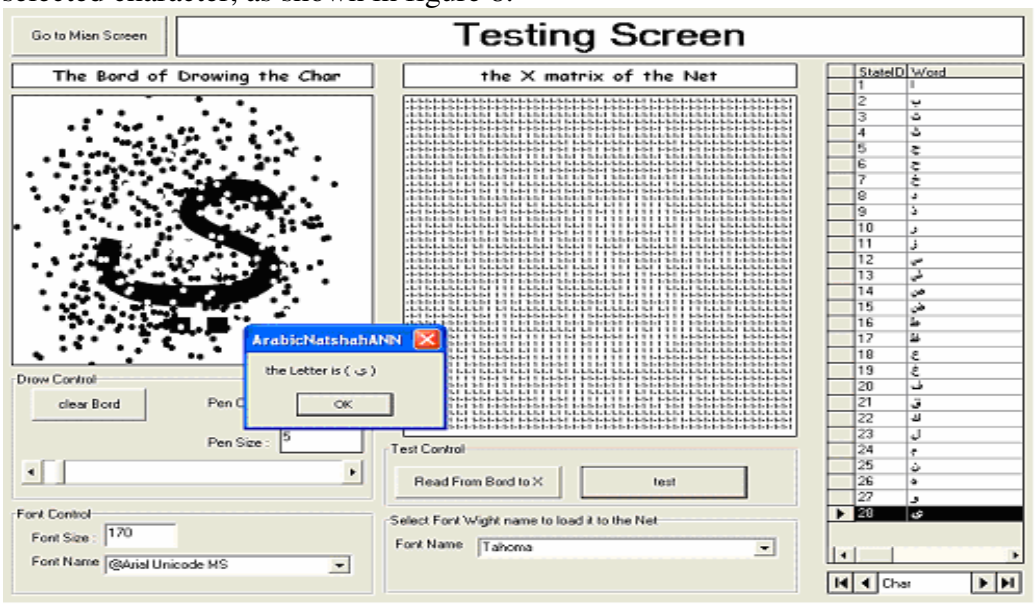

Fig 8.Testing screen with addition of noise.

Another tool is programmed to test for the addition of noise effects on the network performance for all chosen character set, as shown in figure 9. In this screen, you can select the set of characters, its font type, size and the noise percentage required to be added to each input character.

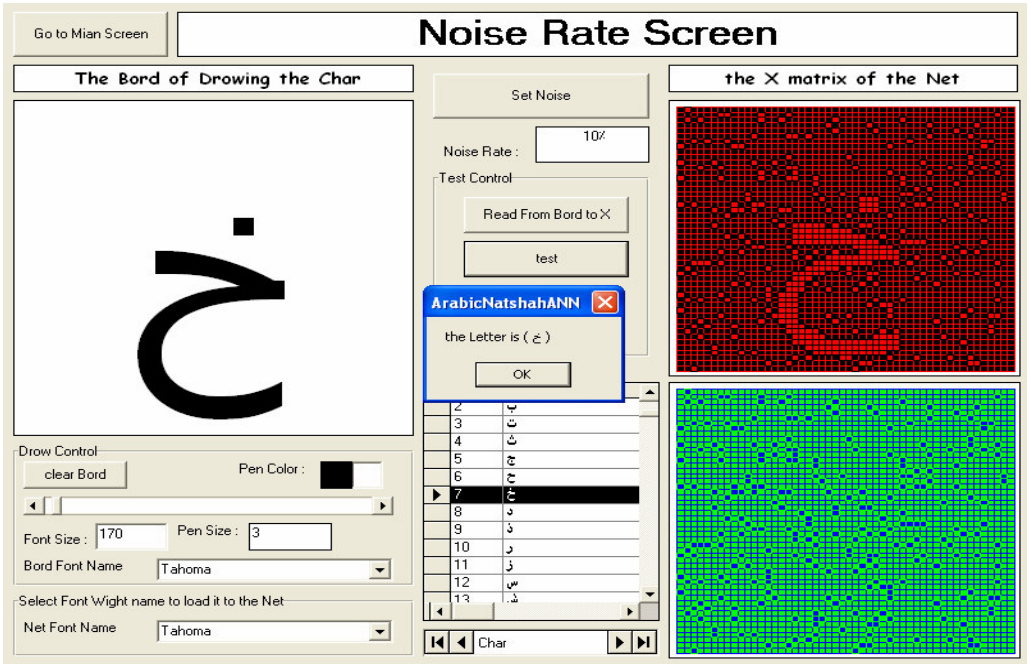

Fig 9. The test screen for noise effect on recognition

Moreover, some other computations can be viewed too. They are testing speed, or the time required to test all the chosen character set, the number of epochs required for acceptable training, the number of patterns correctly recognized, the number of patterns wrongly recognized and the number of rejected patterns.

4. Results

Although, the proposed network is designed to be trained for any character set, it is practically implemented for four of the commonly used fonts. They are (1) Arial (Arabic), (2) Andalus, (3) Simplified Arabic and (4) Tuhama. We have conducted the work on three sets of characters, namely; the basic Arabic alphanumeric set (i.e. 28 basic letters + 10 numerals), the Latin alphabetic set (i.e. 52 letters consisting of small and capital letters) and 10 Arabic numerals only.

4.1 Training and Testing times

Table 1 presents a comparison for the time taken to train the network for the character sets as well as the time required to test the net for the recognition of all the elements of the corresponding set. These measurements are for the four chosen fonts.

Table 1. Training and testing times for the network

Font	**Number of characters**	**Training Time (seconds)**	**Testing Time (seconds)**
Arial (Arabic)	38 Arabic	824	0.041
	52 English	1251	0.043
	10 Number	30	0.033
Andalus	38 Arabic	959	0.040
	52 English	376	0.086
	10 Number	26	0.037
Simplified Arabic	38 Arabic	768	0.044
	52 English	1615	0.041
	10 Number	25	0.036
Tahoma	38 Arabic	1419	0.041
	52 English	335	0.042
	10 Number	25	0.033

Observed training and testing times for the network varies for different fonts. This may

be attributed to the different features found in each set font. Simplified Arabic is found to be the fastest to train among the four sets under study, while testing time has showed no considerable differences.

Numerals set only was much faster to train. This may be attributed to their small number as compared with other sets rather than for any structural differences.

4.2 Error recognition

In order to see how the recognition improves with increasing number of iterations, figure 10 is plotted for the recognition error measureed for repeated training with increasing number of epochs. This network test is only included for the 38 Arabic (Arial) alphanumeric set.

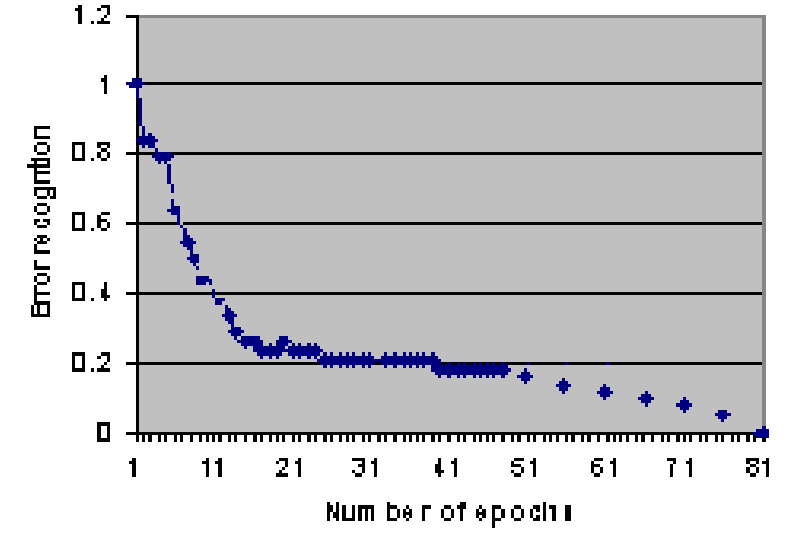

Fig. 10 recognition error measurements

The recognition error decreases exponentially as the number of training epochs increases. We found that the error is extremely reduced after about 80 epochs, which might be considered as fast enough convergence for a practical system.

4.3 Noise effect

Recognition efficiency measurements in the form of correct recognition, error recognition and rejection rates are conducted on the proposed network for the considered fonts. These measurements were done for different level of noise added to the character images in the range from 0 upto 20%. The results are listed in the tables 2-4.

Table 2. Testing rates with no addition of noise.

Font	Set	Recognition Rate	Error Rate	Rejection Rate
Arial (Arabic)	38 Arabic	100%	0%	0%
	52 English	96%	4%	0%
	10 Number	100%	0%	0%
Andalus	38 Arabic	100%	0%	0%
	52 English	100%	0%	0%
	10 Number	100%	0%	0%
Simplified Arabic	38 Arabic	100%	0%	0%
	52 English	96%	4%	0%
	10 Number	100%	0%	0%
Tahoma	38 Arabic	100%	0%	0%
	52 English	100%	0%	0%
	10 Number	100%	0%	0%

Table 3. Testing rates with addition of 10% noise.

Font	Set	Recognition Rate	Error Rate	Rejection Rate
Arial (Arabic)	38 Arabic	79%	21%	0%
	52 English	81%	19%	0%
	10 Number	100%	0%	0%
Andalus	38 Arabic	76%	24%	0%
	52 English	88%	10%	2%
	10 Number	100%	0%	0%
Simplified Arabic	38 Arabic	82%	18%	0%
	52 English	87%	13%	0%
	10 Number	100%	0%	0%
Tahoma	38 Arabic	79%	18%	3%
	52 English	83%	17%	0%
	10 Number	100%	0%	0%

Table 4. Testing rates with addition of 20% noise

Font	Set	Recognition Rate	Error Rate	Rejection Rate
Arial (Arabic)	38 Arabic	53%	47%	0%
	52 English	62%	38%	0%
	10 Number	80%	20%	0%
Andalus	38 Arabic	55%	45%	0%
	52 English	77%	23%	0%
	10 Number	100%	0%	0%
Simplified Arabic	38 Arabic	55%	42%	3%
	52 English	76%	33%	0%
	10 Number	100%	0%	0%
Tahoma	38 Arabic	79%	21%	0%
	52 English	62%	38%	0%
	10 Number	100%	0%	0%

Table 2. Shows that if no noise is added to the image, full recognition is obtained for all Arabic fonts, while 0.96 correct recognition is obtained for the Latin alphabetic set with 0.04 wrong recognition. No character was rejected recognition.

Table 3 included addition of 10% noise level to the image. This has resulted into increased recognition error of up to 24% for Arabic (Andalus) alphanumeric set. However, it is still give full recognition for the numerals.

Table 4 included the addition of 20% noise level to the original images. The results for correct recognition measurement for the letters both Arabic and Latin show drastic deterioration, but those for numerals look unaffected for most fonts. However, there is some error for Arabic (Arial) font.

5. Conclusion

This paper proposes a feed forward neural network that combines properties of perceptron and ADALINE networks. It is tested for implementation of Arabic script classification. It also included the design of a software tool suitable for the training and testing for any character set, together with the effect of adding noise to the original script. The scheme takes the whole character as one feature. The results have shown considerable degree of confidence in character recognition and reliability in a noisy environment. It was evident that when the number of characters increases, the required time for training is increased considerably and the recognition rate decreases especially when noise rate is increased.

To improve the system recognition rate, reliability and noise immunity, the author are working on developing a model that use hidden layers and both feed forward and feedback training scheme.

References

1. Karnofsky, K., "Neural Networks and Character Recognition", Dr. Dobb's Journal, june, 1993.
2. Bishop, C. M., "Neural Networks for Pattern Recognition", Oxford University Press, 2nd Ed., 1999.
3. Caudill, M., "Neural Networks Primer; Part I", Artificial Intelegence Expert, Vol. 2, No. 12, Dec. 1987.
4. Chaltra, S., "Use Neural Networks for Problem Solving", Chemical Engineering Progress, April 1993.
5. Cho, S, "Neural Network Classifiers for Recognizing Totally Unconstrained Hand Written Numerals", IEEE Trans. on Neural Networks, Vol. 8, No. 1, Jan. 1997.
6. Haykin, S., "Neural Networks: A Comprehensive Foundation", Prentice Hall, 1999.
7. Wasserman, P. "Neural Compution: Theory and Practice", Van Nostrand Reinhold International Company Limited, 1989.
8. Davalo, E. and Nain, P., "Neural Networks", Macmillan Education Limited, 1991.
9. Lippman, R., "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, April 1987.
10. Tank, D. and Hopfield, J., "Collective Computation in Neural-like Circuits", Scientific American, Dec. 1987.
11. Mahmmod, M. A. B., "An Equivalent Model as Medium Scale Neocognitron-Like Brain Model", M.Sc. Thesis in Electrical Engineering, University of Basrah, Iraq, 1999.
12. Ali, H. A. and Mahmmod, M. A. B., "A Hybrid Neural Network Model for Character Recognition of Hand Written Numerals", Journal of the Association of the Advancement of Modeling and Simulation Techniques in Enterprises (AMSE), Vol. 45, No. 2, 2002.
13. Ali, H. A. and Mahmmod, M. A. B., "Feature Contents Study of Arabic Numerals Towards Automatic Recognition Model using Combined Neural Network Technique", Accepted ACIT 2002 conference, Qatar, 16-19th Dec. 2002.
14. Gouhara, K.Imai, K. and Uchikawa, Y., "Position and size Representations by Neural Networks", Control and Computers, Vol. 17, No.