Comparison of Neural Network Parameters for Classification of Arabic Handwritten Isolated Characters

Nidal Lamghari
FSTG, LAMAI
Cadi Ayyad University
Marrakesh, Morocco

My El Hassan Charaf
Faculty of sciences, ISO-LAB
Ibn Tofail University
Kenitra, Morocco

Said Raghay
FSTG, LAMAI
Cadi Ayyad University
Marrakesh, Morocco

ABSTRACT
The neural network is a static classifier that requires fixed-size vector functions. For this reason, it is considered as a very effective approach for recognizing characters and graphemes. More than 80% of the research that implements neural networks uses backpropagation. The retro-propagating neural network can be used in many applications such as character recognition, face recognition, etc. Training of neural networks is a complex task in the field of supervised research. The main difficulty is to find the most appropriate combination of network architecture, learning function, transfer and training for the classification task. In this paper we dress the recognition of Arabic handwriting isolated characters using two types of neural networks: a feed forward and a cascade forward. We achieve different experiments by varying the number of hidden layer neurons, learning functions, and transfer functions. For that, we use our database for Arabic handwritten characters and ligatures (DBAHCL) in the training, test and validation phases. We compare the results based on the mean squared error, accuracy, convergence rate, and classification accuracy.

General Term
Pattern Recognition, classification, neural network.

Keywords
Handwritten Arabic characters, recognition, DBAHCL, neural network, transfer function, learning function, feed forward, cascade forward.

1. INTRODUCTION
The problem of reliable handwriting recognition is always a major challenge. This is an active area of research in pattern recognition. For the time being, and although research in this area has been going on for more than 30 years, the problem of automatic cursive recognition is still unresolved. However, it seems that the recognition of cursive writing has an important role to play in future recognition systems such as postal address and postal code recognition, the processing of forms, the automatic processing of bank checks, etc. Therefore, this area of research remains a relevant field.

Currently, the problems of recognizing Latin handwriting are partially solved, and the automatic reading of print has made great progress in many areas. Research in this area is moving towards the analysis of much less demanding documents than those treated so far. Unlike Latin, the recognition of Arabic handwriting still remains at the level of research and experimentation, the problem is not yet solved even if in some applications with limited vocabularies and mono-cast, some appreciable results are reported. The work is generally focused on the development methodology rather than the realization of a marketable final product, which's still unreal.

The delay of Arabic writing in relation to Latin writing can be attributed to the morphological complexity of the Arabic alphabet. In addition, the lack of common validation and testing protocols greatly contributes to this delay. Research on the recognition of Arabic handwriting dates back to the 1980s. Since then, research has proliferated in this area. In recent decades, the recognition of handwritten Arabic characters has taken a new life. Indeed, several approaches and methods have been proposed by the researche*rs, in order to improve the recognition rates of the recognition systems of handwritten Arabic characters. However, the research remains small compared to that carried out in Latin script. Inspired by the functioning of the human nervous system, neural networks is actually one of the most effective tools for the classification of Arabic manuscript characters. In this paper, we present a comparative study of the different architectures of neural networks used in the classification of Arabic manuscript characters. We conduct different experiments by varying the number of hidden layer neurons, learning functions, and transfer functions. We also compare the results obtained with normalized data and those obtained using non-normalized data.

2. RELATED WORK
The recognition of Arabic handwriting has been approached in most works with the same approaches as for other scripts. However, its semi-cursive nature and its morphological particularities sometimes require special treatments. In this section, we present some of these systems, focusing on the characteristics used, the classifier used, the test results obtained and the database used.

In [1] and [2], the authors propose a system of recognition of Arabic manuscript words, using hidden Markov models (HMMs). First, the words are segmented and normalized. Segmentation is implicit using sliding windows. During the moving of the sliding window, structural features are extracted such as the number of pseudo words and diacritics. The authors combine these characteristics with average pixel intensity, discrete cosine transform (DCT) coefficients, and invariant moments. Basically, intensity characteristics and other statistical characteristics are used to train HMM. Structural features are used for reclassification which, according to the authors, gives better accuracy. In-depth experiments were used with the four versions of the IFN / ENIT database. Their system achieved a recognition rate of 89.24%.

Lawgali, et al., [3] compare the use of the Discrete Cosine Transform (DCT) with the discrete wavelet transform (DWT) to extract the characteristics used to recognize Arabic handwritten characters. The experiments were carried out in two stages. The first step was applied to a database containing 1600 isolated forms of Arabic characters. The second

experiment was conducted using 5600 Arabic manuscript characters that covered all forms of Arabic characters. The analysis of their results showed that the extraction of the characteristics by DCT gives better results. A recognition rate of 94.87% was achieved in the first experiment on a sample of 1600 images of size 64x64. For the second experiment, the researchers obtained a rate of 79.87%.

Farah Hanna Zawaideh uses neural networks in [4]. The method proposed by the author consists in dividing the image of the character in 6 by 4 segments. The size of the image being 48 * 32, the size of the segment or grapheme is then 8 * 8 pixels. The author used 120 characteristics for each grapheme. It is essentially the ratio between the white pixels and the black pixels, the average of the spatial segment, the variance of the two most distant vertical pixels, the average value of the line and the total variance of the pixels of the segment. About a hundred characters segmented into graphemes have been classified by a cascade neural network. The author has designed an architecture consisting of 4 neural networks. The first and main network is a multi perceptrons. The output of this network is communicated to 3 other linear vector quantizer networks (Linear-Vector Quantizer). The system achieved an average recognition rate of 68.10%.

In 2013, Al Hamad [5] uses four different neural models: Feed-Forward Back-Propagation (FFBP), Multilayer Perceptron (MLP), Radial-Basis Function (RBF) and Self-Organizing Map (SOM). The database used in this research was obtained from twenty different people: 620 characters written by the first 10 people are used for training. And for the test, 500 words were randomly extracted from two paragraphs containing all possible forms of Arabic characters written by the other ten people. Firstly, the proposed approach detects and traces the character outline. Then, the directions of the line segments comprising the characters that are detected and the foreground pixels are replaced by appropriate direction values. Finally, feature characteristics based on the location of background transitions are extracted and neural learning and classification are performed. The character recognition rate reached 72.58% using a Perceptron Multi Layer (MLP), 95.32% using the RBF network, 78.06% for the FFBP and 24.35% using the SOM network.

In [6], the authors describe an algorithm for segmenting cursive words into graphemes. This algorithm defines specific characteristic points identified in the skeleton of the word and operates in two steps. In the first, a baseline is estimated using the projection histogram algorithm. Then, based on the outline, the authors proceed to the detection of the secondary components (diacritics, etc.). In the second step, the word is segmented into pseudo words, then the pseudo-words into graphemes. Segmentation in graphemes is based on the detection of characteristic points extracted from the skeleton such as: end points, branch points and cross points. Additional features are extracted in the feature extraction step. It consists of 103 characteristics classified in six classes: statistical characteristics, configuration characteristics, skeletal characteristics, contour, elliptical Fourier descriptor and directional characteristics. There are 15 statistical characteristics extracted from the binary image of the word to be recognized, notably: the area, the width of the word and its height, the pixel density, the invariant moments, the angle of orientation of the word. The configuration characteristics are extracted from the outline of the object and its environment. Compared to the baseline of the word, the authors extract three characteristics: the proportion of pixels above the baseline, the distance between the center of the object and the baseline and the distance between the black pixel higher and the baseline. The other configuration characteristics are essentially: the existence or not of loops, the number of secondary components, and their position. Four other features were extracted from the skeleton. This is the number of branch points in the skeleton of the object, the number of endpoints, and two experimental features calculated as the sum of the moments of the bisection angles of the body edge points. The outline features used in their study are: the number of boundary pixels, the perimeter length, the perimeter / diagonal ratio, and the compactness ratio. Directional features are extracted from the string codes of the object's outline.

The use of neural networks resulted in a reduction of the error rate of 18.5% and a reduction in execution time of 31% compared to the MDLSTM system.

Abed et al. [7] proposed a system for recognizing isolated Arabic handwritten characters based on neural networks and the algorithm for the retro propagation of error. This system is for the following 12 characters: (ا, ي, و, م, ك, ف, غ, ط, ش, ت, ج, ذ) because some characters have the same main body and a location and / or different number of diacritics. The authors used 20 images of the 12 characters for training and 20 others for the test. A recognition rate of 93.61% was obtained with an average training time of 36.18 seconds. The neural network used in their study consists of an input layer of 252 neurons, three hidden layers and an output layer of 12 neurons. For feature extraction, the authors use the zoning technique. Since the 18x18 size binary image is adjusted in a matrix, it is divided into nine 6x6 square sub-matrices.

Omer Balola et al. [8] [9], present two systems for the recognition of isolated Arabic handwritten characters. The database used in their searches consists of 30,600 character images, divided into 23,800 images for training and 6800 images for testing. The characteristics extracted in the two experiments are essentially: density, analysis of principal and secondary components, and other structural features. In the first system [8], the authors proceed in two stages. In the first stage, they use a public classifier to process all the characters and classify them according to characteristics in fifteen different groups. The second step is to assign each group a classifier that the authors call a private classifier. In their research, the authors use a neuron network with an input layer of 100 neurons, a hidden layer of 600 neurons and an output layer of 34 neurons. This system achieved a 78.77% recognition rate for testing data using a single overall step and 92.77% applying two steps. As for the second system [9], which looks like the first one, it is based on a system of adaptive inference neuro-fuzzy. Using a single classifier for all test data, the system achieved a recognition rate of 96.2%. On the other hand, in two steps, the system achieved a 99.5% recognition rate for testing data set. This rate was obtained using a private classifier and then a neuro-fuzzy was created for each group to recognize and classify the characters of a group.

In [10], the author proposes a hybrid classification of isolated Arabic handwritten characters. The first classifier is the SVM, which classifies characters into two groups: characters with point (s) and characters without points. This step allows, according to the author, to facilitate the task to the neural network which is the second classifier. The experiment was performed using a sample from the IFN / ENIT database: 2049 images were used for training, 439 images for the test and 439 images for validation. DWT and wavelet are used to

extract the characteristics of binarized character images. The wavelet characteristics are used to find the low and high pixel density frequency in the character and to detect in which region of the area are the dots. On the one hand, the DWT logically divides the image into four parts to locate the dotted lines. On the other hand, the curvelet is used to study the shape, the main body and the direction of each continuity of the character. Different arrangements occur depending on the number of objects in each image and its location. For example, every letter without a point counts as a single object, whereas there are characters with two, three, or four objects. The images are then processed by the SVM which produces an output equal to 2 if the character does not contain a point and an output equal to 1 if it contains one or more points. Then the character is treated by a network of neurons. The SVM classifier provides 92.2% accuracy by dividing the dataset into two groups. And the maximum overall recognition rate obtained is 99%.

Finally, we propose in a previous work [11] a system of recognition of Arabic handwritten characters using the command nprtool a network of neurons feed forward of a hidden layer of 70 neurons. The input layer consists of 66 neurons. Thus, we implement five methods to produce a hybrid vector of 66 structural, statistical and regional characteristics. The structural features we use are essentially Freeman's code, the nature, number and position of the diacritics and are presented in our article [12]. Experimental results revealed a very good recognition rate for isolated characters from our DBAHCL database [13]. Indeed, by assigning 3400 isolated characters of this database by our experience, we obtained 98.27% recognition rate.

3. PROPOSED METHODOLOGY

The traditional organization of a handwritten character recognition system goes through four main and classic steps: acquisition, pre-processing, feature extraction, classification and recognition. Since the operation of pre-treatment has become a classic tool in the field of image processing, we do not present it in this study. For feature extraction we use the methods described in [11] to generate hybrid vectors of 66 characteristics. And to classify and recognize the handwritten characters, we use different neural network architectures using the MATLAB neural network toolbox (nntool).

3.1 Dataset definition

The first step in a recognition system based on supervised learning models is the preparation and collection of data. The data collection and definition process consists of describing the type of data used in training, testing and validation. The database used is our database DBAHCL presented in [13]. This database is designed to cover all forms of Arabic characters, including ligatures. It contains 9,900 ligatures and 5500 characters written by 50 writers. In this work, we are interested only in isolated characters. The table below shows the characters supported by our system with their outputs.

Table 1. All isolated Arabic characters concerned by our proposed system

7	6	5	4	3	2	1
ك	د	س	ر	ل	ع	ح
14	13	12	11	10	9	8
خ	ا	م	و	ص	ط	ه
21	20	19	18	17	16	15
ض	ظ	ف	ذ	ز	ن	غ
28	27	26	25	24	23	22
آ	إ	ئ	ؤ	أ	ج	ب
	34	33	32	31	30	29
	ث	ش	ي	ت	ق	ة

3.2 Experimentation and results

The second step is to identify the parameters to use for the prediction models. To obtain different results, we implement the inverse propagation algorithm with Neural Network Tool (NNTOOL) which allows to import, create and use different architectures of neural networks.

Matlab's nntool graphical tool has a relatively complex hierarchical structure and allows exploiting and testing different network architectures, different training, learning and transfer functions. This tool provides predefined algorithms and allows creating, training, visualizing and simulating neural networks. It supports both types of learning: supervised and unsupervised by offering multiple architectures such as multilayer perceptron, recurrent networks, radial-based networks, and more. In our experiments, we used the Trainlm training function for the hidden layer.

The implementation of a neural network model from the nntool interface consists of three phases: training, test and validation. The data is divided into three parts: 70% of the isolated characters from our database DBAHCL are used for training (0.7 * 3400), 15% for the test phase and 15% for the validation.

The training phase makes it possible to determine the network connection parameters using the optimization technique. The test phase consists of checking the network setting on the data not used in the training phase. The validation phase, on the other hand, is performed on the last part of the data and is used to measure the generalization of the network.

We use two different types of neural networks: a feed forward and a cascade forward. We conduct different experiments by varying the number of hidden layer neurons, learning functions, and transfer functions. Also, we compare the results obtained with standardized data and those obtained using non-standardized data. All the networks used consist of an input layer, an output layer and a hidden layer.

We first tested with non normalized data starting with a hidden layer of 10 neurons and incremented this number each time. The same experiments were conducted using normalized data but with the trainscg training function. The Trainlm function could not be used with normalized data because of the memory space. Tables 3 and 4 below show these different experiences.

Table 2. Results obtained with a feed forward

Transfer function of the hidden layer	Number of neurons	Non normalized data		Normalized data	
		MSE with Learngd	MSE with Learngdm	MSE with Learngd	MSE with Learngdm
Tansig	10	0.122	0.0395	$5.05e^{-03}$	0.0144
	20	0.22	0.106	$1.6e^{-03}$	$3.5e^{-03}$
	30	0.307	0.266	$2.9e^{-03}$	$1.2e^{-02}$
	40	0.436	0.155	$5.06\ e^{-03}$	**$2e^{-03}$**
	50	**$3.75e^{-04}$**	0.0585	$2.9e^{-03}$	$2.1e^{-02}$
	60	0.0324	$5.7\ e^{-03}$	**$3.74\ e^{-04}$**	$8.6e^{-03}$
	70	0.0186	**$9.11e^{-06}$**	$3.12\ e^{-03}$	$9.6e^{-03}$
Logsig	10	0.0794	0.0807	0.0116	0.0154
	20	0.842	1.11	0.0135	0.00495
	30	0.103	0.195	0.0202	**$3.14e^{-03}$**
	40	0.124	0.0190	0.00566	0.0106
	50	**0.0114**	0.244	0.0178	0.0201
	60	0.575	**0.0100**	0.0139	0.00724
	70	0.131	0.0891	**$3.18e^{-03}$**	0.00902
Purelin	10	4.53	4.20	0.569	**0.112**
	20	4.25	4.42	0.196	0.570
	30	5.69	4.42	0.294	0.287
	40	4.44	4.40	0.288	0.570
	50	4.52	4.32	0.295	0.291
	60	4.49	4.40	0.294	0.275
	70	4.26	4.31	0.112	0.295

Table 3. Results obtained with a cascade forward

Transfer function of the hidden layer	Number of neurons	Non normalized data		Normalized data	
		MSE with Learngd	MSE with Learngdm	MSE with Learngd	MSE with Learngdm
Tansig	10	0.268	0.470	$1.54e^{-02}$	$2.28e^{-02}$
	20	1.13	0.537	$2.47e^{-02}$	e^{-02}
	30	3.68	3.65	$1.77e^{-02}$	$1.95e^{-02}$
	40	0.206	0.221	**$5.74e^{-03}$**	$1.95e^{-02}$
	50	0.566	0.241	$2.03e^{-02}$	$1.46e^{-02}$
	60	0.00338	$1.65e^{-02}$	$2.80e^{-02}$	$1.52e^{-02}$
	70	0.00817	**$2.84\ e^{-21}$**	$1.87e^{-02}$	$1.22e^{-02}$
Logsig	10	2.73	0.146	$6.09e^{-03}$	$6.44e^{-03}$
	20	1.50	0.609	$6.97e^{-04}$	$2.38e^{-02}$
	30	3.79	0.792	$2.02e^{-02}$	$2.26e^{-02}$
	40	0.387	0.116	$2.55e^{-02}$	$8.31e^{-03}$
	50	0.0380	$4.03e^{-02}$	$2.80e^{-02}$	$1.99e^{-02}$
	60	0.0708	$8.28e^{-02}$	$2.24e^{-02}$	$1.24e^{-03}$
	70	0.0847	$4.04e^{-02}$	$2.19e^{-02}$	**$5.61e^{-04}$**
Purelin	10	4.38	4.49	$2.94e^{-02}$	$2.94e^{-02}$
	20	4.33	4.34	$2.87e^{-02}$	$5.16e^{-02}$
	30	4.55	4.40	$2.72e^{-02}$	**$2.45e^{-02}$**
	40	4.37	4.44	$2.85e^{-02}$	$2.63e^{-02}$
	50	4.32	4.23	$2.63e^{-02}$	$2.79e^{-02}$
	60	4.31	4.21	$2.88e^{-02}$	$2.68e^{-02}$
	70	4.49	4.34	$2.86e^{-02}$	$2.70e^{-02}$

4. DISCUSSION OF RESULTS

According to the different experiments carried out with the two tools nprtool in [11] and nntool in this paper, we find that the tool nntool is richer than the tool nprtool. Indeed, with the tool nntool, we can try different architectures by varying the transfer functions and compare the results. Also, we can notice, on the one hand, that the transfer functions of the Tansig and Logsig hidden layer provide better results in our case than the Purelin function. On the other hand, we find that the normalization of the data makes it possible to stabilize the different networks of neurons used. Certainly, it makes it possible to minimize the mean squared error (mse). However, to evaluate the performance of a neural network, it is not enough to evaluate the convergence of its quadratic errors, but also the recognition rate and the execution time. The average execution time of all neural networks tested with normalized data was in the range of 02 min 35 s and 02 min 49 s using non-normalized data. In spite of that, the recognition rates obtained using normalized data remain modest compared to those obtained with non-normalized data. In the tables and figures below, we compare the recognition rates obtained using the two types of network: feed forward and forward cascade. Looking at the table 4 below, we note that the recognition rates for a single feed forward networks using normalized data vary between 10.85% and 98.65%. This last rate exceeds the other obtained with the tool nprtool. The best rate was 98.27% by using the latter tool that requires normalization of data. This result was obtained by training twice a network of neurons feed forward 70 neurons in the hidden layer. While using the nntool tool, we were able to record a recognition rate of 98.65% by training a single feed forward network of 60 neurons in the hidden layer. Recognition rates obtained with non-normalized data are larger; they vary between 82.72% and 99.90%. The same observation was made using a forward cascade neural network. In fact, the recognition rates obtained with this type of network vary between 16% and 98.02% for normalized data and between 73.56% and 99.91% for non-normalized data (Table 5 below). In Tables 4 and 5, ‘GD’ and ‘GDM’ represent the Learngd and Learngdm functions respectively.Data normalization is typically used to optimize a network from a state that converges more quickly. It has been empirically demonstrated in many research that data

normalization can accelerate learning and improve the rate of convergence, but there is no guarantee for this. In our case, the normalization of data makes the convergence rate slower. In our opinion, this can be due to the diversity of characteristics used. In effect, normalization destroys this diversity and we lose information.

Table 4. Rate of recognition obtained by a Feed Forward

Neurons	Tansig				Logsig				Pureline			
	Normalized		Non normalized		Normalized		Non normalized		Normalized		Non normalized	
	GD	GDM	GD	GDM	GD	GDM	GD	GDM	GD	GDM	GD	GDM
10	89.58%	81.00%	**99.78%**	**99.79%**	76.27%	68.43%	**99.89%**	**99.80%**	12.45%	10.85%	**90.30%**	**88.93%**
20	95.80%	92.77%	**99.48%**	**99.67%**	70.75%	90.17%	**99.58%**	**98.57%**	10.84%	14.63%	**82.72%**	**91.02%**
30	93.56%	91.25%	**99.59%**	**99.43%**	54.93%	93.27%	**99.68%**	**99.59%**	25.74%	28.40%	**87.61%**	**91.04%**
40	90.21%	95%	**99.72%**	**99.73%**	88.89%	78.02%	**99.66%**	**99.84%**	14.32%	15.24%	**91.24%**	**91.29%**
50	93.49%	93.45%	**99.89%**	**98.57%**	61.76%	94.02%	**99.85%**	**99.71%**	16%	14.85%	**90.46%**	**91.35%**
60	98.65%	83%	**99.29%**	**99.84%**	70.99%	85.42%	**99.62%**	**99.86%**	17.52%	19.23%	**90.51%**	**91.21%**
70	93.42%	80%	**98.77%**	**99.90%**	93.59%	82.45%	**99.60%**	**99.74%**	14.85%	10.78%	**89.12%**	**91.36%**

Table 5. Rate of recognition obtained by a Cascade Forward

Neurons	Tansig				Logsig				Pureline			
	Normalized		Non normalized		Normalized		Non normalized		Normalized		Non normalized	
	GD	GDM	GD	GDM	GD	GDM	GD	GDM	GD	GDM	GD	GDM
10	67.44%	47%	**99.70%**	**99.70%**	87.63%	86.52%	**99.40%**	**99.82%**	40%	20%	**90.59%**	**89.12%**
20	40.54%	79.41%	**92.10%**	**99.62%**	97.35%	42.38%	**99.18%**	**99.12%**	16%	16.05%	**91.41%**	**91.61%**
30	62.62%	56.89%	**97.39%**	**97.41%**	54.76%	46.95%	**96.54%**	**99.04%**	26%	40%	**80.67%**	**91.59%**
40	88.14%	56.75%	**99.79%**	**99.76%**	35.82%	82.97%	**99.21%**	**99.67%**	16.90%	32.04%	**91.60%**	**92.29%**
50	54.60%	68.44%	**99.56%**	**99.73%**	21.26%	55.02%	**99.61%**	**99.85%**	31.08%	22.79%	**90.65%**	**91.80%**
60	61.07%	67.58%	**99.82%**	**99.87%**	47.08%	96.03%	**99.72%**	**99.87%**	14.10%	29.10%	**91.25%**	**91.45%**
70	59.12%	75.92%	**99.80%**	**99.91%**	49%	98.02%	**99.68%**	**99.79%**	16.37%	27.91%	**73.56%**	**91.56%**

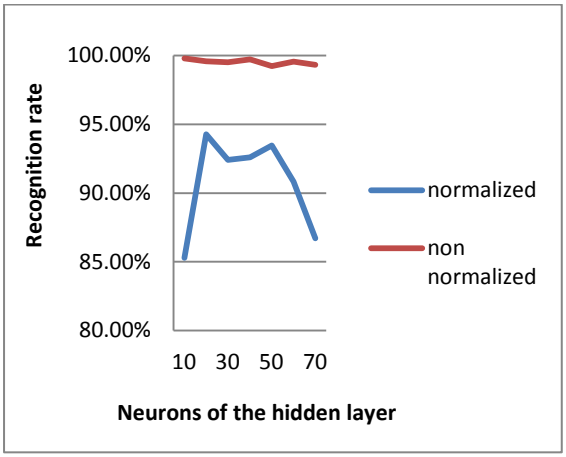

Fig 1: Rates obtained using the Tansig transfer function and a feed forward

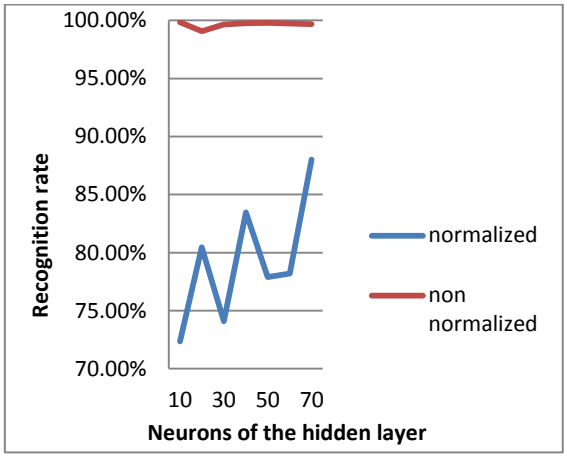

Fig 2: Rates obtained using the Logsig transfer function and a feed forward

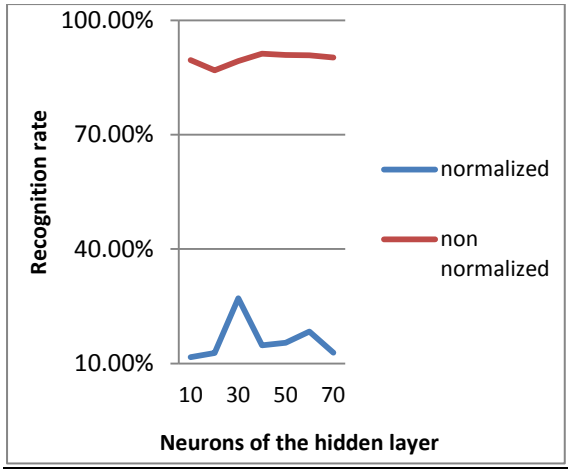

Fig 3: Rates obtained using the Purelin transfer function and a feed forward

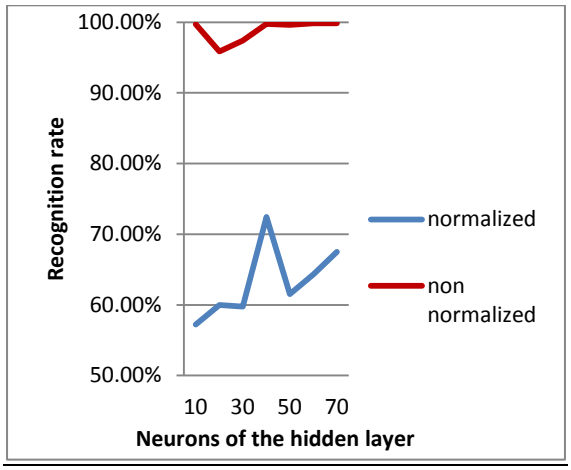

Fig 4: Rates obtained using the Tansig transfer function and a cascade forward

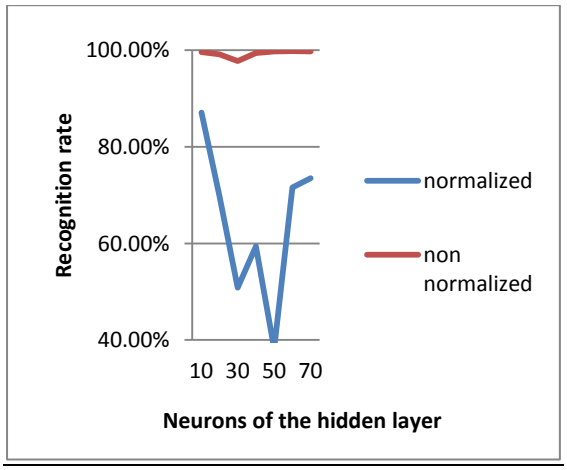

Fig 5: Rates obtained using the Logsig transfer function and a cascade forward

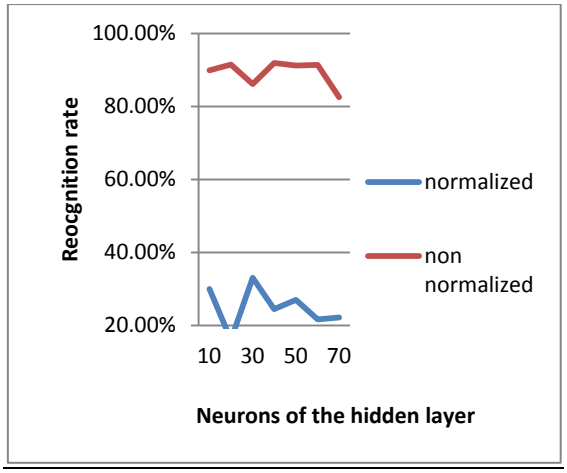

Fig 6: Rates obtained using the Purelin transfer function and a cascade forward

The best recognition rate is 99.91%. It was obtained by using a forward cascade network of a single hidden layer of 70 neurons whose transfer function is Tansig. Indeed, the quadratic error obtained with this network is $2.84e^{-21}$. By training this network once, we managed to have a 100% recognition rate of the training data, a 99.73% as a result for the recognition of test data and a recognition rate of 99.81% for validation data such a very important and encouraging number. However, to give a rough estimate of the performance of our system and for completeness, we compare our system to the best state-of-the-art systems that use roughly the same type of characteristics and the same classifier. In the next section, we present a brief comparison of our system with other Arabic handwriting recognition systems. Most systems have been applied using different databases.

5. COMPARISON WITH OTHER SYSTEMS

Compared to related work, we find that the system presented in [10] achieved a recognition rate of 99%. Their system combines neural networks and SVMs. These allow classifying the isolated Arabic manuscript characters according to whether or not they contain diacritic points. A neural network is then used to classify the characters. The recognition rate obtained in their study is lower than the rate we obtained.

The Balola [9] neuro-fuzzy adaptive inference system achieved a very good recognition rate 99.5%. This rate was obtained by combining two classifiers. The feature vector used consists of 100 features that are essentially: density, principal and minor component analysis, and other structural features. The adopted neural network includes a hidden layer of 600 neurons. However, the number of neurons of the hidden layer seems too important, which risks reducing the capacity of generalization of the network and implying over-learning. We can judge at this point that our system is more stable and reliable since the characteristics used are practically the same and the number of neurons of the hidden layer is smaller.

The system of recognition of the isolated handwriting Arabic characters proposed in [7] achieved a recognition rate of 93.61%. This system concerns 12 characters and is based on a feature vector of 252 primitives extracted using the zoning technique. It relies on a neural network of 3 hidden layers. However, the training and test databases contain 20 images

each. This makes their work very modest compared to the different systems of literature.

In addition, the work done by Al Hamad [5] which combines local characteristics and global structural information gives an average recognition rate of 95.32% using a radial network of basic functions (Neural Radial-Basis).

The Zawaideh system [4] presents a feature extraction approach to achieve high recognition accuracy of handwritten Arabic letters. This approach is based on statistical characteristics such as: the ratio of the white pixels to the black pixels, the average of the spatial segment, the variance of the two most distant vertical pixels, the average value of the line and the total variance of the segment's pixels. The system achieved a recognition rate of 68.10%, which remains very modest compared to rates obtained by other systems.

Lawgali's system [3] achieved a recognition rate of 87.08% using the Cosines Discrete Transform (DCT) with artificial neural network (ANN).

However, our system achieved a recognition rate of 98.27% using the nprtool tool, a recognition rate of 99.90% using a feed forward via the nntool tool and 99.91% by adopting a cascade forward also by using the same tool. Indeed, the rate has increased compared to the template matching presented in our article [12]. The improvements were achieved by combining five feature extraction methods: resizing, calculating the pixel density of the 4 image blocks, and structural features that include the nature, number, and position of the diacritics combined with other structural characteristics and the standardized Freeman code, the seven invariant moments and four regional characteristics (eccentricity, extent, solidity, and orientation). The results obtained show that the combination of statistical, regional and structural characteristics gives very well results in the classification of Arabic handwriting characters. It also confirms the importance of structural features, especially diacritics. Finally, we can say that the rates obtained with our system are among the best rates. Nevertheless, these rates can be significantly improved by using deep learning.

6. CONCLUSION

Neural networks can be used effectively in the classification and recognition of Arabic isolated handwritten characters with an appropriate combination of learning, transfer and training functions. In the proposed work, we tried two different architectures of neural networks by varying the number of layer neurons, transfer and learning functions. We noticed that the transfer function 'Tansig' and the learning function 'Learngdm' provide very good results. We also found that in our case, data normalization makes the convergence rate slower. This is due, in our opinion, to the diversity of characteristics used. Indeed, normalization destroys this diversity and we lose information.

7. ACKNOWLEDGMENTS

I acknowledge the support provided by my two supervisors Pr. Said Raghay and Pr. My El Hassan Charaf and the members of the Laboratory of Applied Mathematics and Computer Science, Faculty of Science and Techniques, Cadi Ayyad University, Marrakesh, Morocco and the members of the laboratory of Informatics, Systems and Optimization "ISO-LAB" Faculty of Science, Ibn Tofail University Kenitra, Morocco.

8. REFERENCES

[1] Alkhateeb, J. 2010. Word Based Off-line Handwritten Arabic Classification and Recognition. Doctoral Thesis. School of Computing, Informatics and Media, University of Bradford.Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] AlKhateeb, J. and Jawad, H. and al. 2011. Offline handwritten Arabic cursive text recognition using Hidden Markov Models and re-ranking. Pattern Recognition Letters 32.8 : 1081-1088.

[3] Lawgali, A., Bouridane, A., Angelova, M. and Ghassemlooy, Z. 2011. Handwritten Arabic character recognition: Which feature extraction method? International Journal of Advanced Science and Technology, 34, 1-8.

[4] Zawaideh, F.H. 2012. Arabic Hand Written Character Recognition Using Modified Multi - Neural Network", Journal of Emerging Trends in Computing and Information Sciences, Vol. 3, NO, ISSN 2079 -8407, 7.

[5] Al Hamad, H.A. 2013. Use an efficient neural network to improve the Arabic handwriting recognition. Signal and Image Processing Applications (ICSIPA), 2013 IEEE International Conference on. IEEE.

[6] Abandah, Gheith, A., Fuad, T., Jamour, T. and Esam, A. 2014. Recognizing handwritten Arabic words using grapheme segmentation and recurrent neural networks. International Journal on Document Analysis and Recognition (IJDAR) 17.3: 275-291.

[7] Abed, Majida, A. and Hamid Ali Abed Alasad, A. 2015 High Accuracy Arabic Handwritten Characters Recognition Using Error Back Propagation Artificial Neural Networks. International Journal of Advanced Computer Science and Applications 6.2.

[8] Balola, O., Shaout, A. and Elhafiz, M. 2015. Two stage classifier for Arabic Handwritten Character Recognition. International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 12.

[9] Balola, O. and Shaout, A. 2016. Hybrid Arabic Handwritten Character Recognition Using PCA and ANFIS. International Arab Conference on Information Technology.

[10] Al-Jubouri, M. A. H. 2017. Offline Arabic Handwritten Isolated Character Recognition System Using Support vector Machine and Neural Network. Journal of Theoretical & Applied Information Technology 95.10.

[11] Lamghari, N., Charaf, M.E.H, Raghay, S. 2017 Hybrid Feature Vector for the Recognition of Arabic Handwritten Characters Using Feed-Forward Neural Network. Arabian Journal for Science and Engineering, p. 1-9. Doi:10.1007/s13369-017-2969-1.

[12] Lamghari, N., Charaf, M.E.H, Raghay, S. 2016. Template Matching for recognition of handwritten Arabic characters using structural characteristics and Freeman code. The International Journal of Computer Science and Information Security 14(12), p.31.

[13] Lamghari, N. and Raghay, S. 2017. DBAHCL: Database for Arabic handwritten characters and ligatures. Int J Multimed Info Retr. Doi:10.1007/s13735-017-0127-x.