Acta Polytechnica **58**(5):297–307, 2018
doi:10.14311/AP.2018.58.0297

available online at http://ojs.cvut.cz/ojs/index.php/ap

NEW APPROACH FOR ONLINE ARABIC MANUSCRIPT RECONGNITION BY DEEP BELIEF NETWORK

Benbakreti Samir*, Aoued Boukelif

University of Djillali Liabes, Department of Electronic, Sidi Bel Abbes 22000, Algeria

* corresponding author: benbakretisamir@gmail.com

Abstract. In this paper, we present a neural approach for an unconstrained Arabic manuscript recognition using the online writing signal rather than images. First, we build the database which contains 2800 characters and 4800 words collected from 20 different handwritings. Thereafter, we will perform the pretreatment, feature extraction and classification phases, respectively. The use of a classical neural network methods has been beneficial for the character recognition, but revealed some limitations for the recognition rate of Arabic words. To remedy this, we used a deep learning through the Deep Belief Network (DBN) that resulted in a 97.08 % success rate of recognition for Arabic words.

Keywords: Arabic manuscript; online recognition; neural networks; MLP; TDNN; RBF; deep learning; DBN.

1. Introduction

Nowadays, the cursive writing recognition problem represents a major challenge for both handwritten and typed forms. The complexity of this problem comes from the style, the variability of the patterns, the tilt in the case of the unconstrained handwriting, this is even worse for some languages that present more complicated morphologic characteristics. The Arabic handwriting, which is cursive by nature, presents a big variability and is consequently very difficult to resolve. The recent electronic revolution allows the emergence of new typing devices capable of creating high quality online documents, such as exam papers, notes, filling forms, online reporting, etc. This progress extends the application field of the online handwriting recognition, which was restricted by terminals of small size (PDA, Smartphone) that support character recognition only. The online documents represent a new source of information that has few applications in the field of the pattern recognition. They are represented by signals which may be assimilated to sampled trajectories captured from the handwriting instrument as a temporal sequence of points $(x(t), y(t))$, represented in an orthonormal coordinate system. The Arabic script is semi-cursive in the two forms, printed and handwritten. The word may be composed of one or several pseudo-words. These pseudo-words are ligated horizontally or vertically, which makes the segmentation task very complicated. The shape of the character differs according to its position in the word. Some characters include diacritical points, which lie above or below. However, others have the same body or shape, only the diacritical point allows to differentiate between two characters. Also, in the handwritten Arabic text, the variations in the horizontal bands, according to the calligraphy of the characters contained in the pseudo-word, appears [1]. In the case of the manuscript, this complexity is more important, because there are other problems that may arise, such as: the variability intra and inter-writer, the overlap of pseudo-words, the fusion of diacritical points, the writing conditions or the writer state.

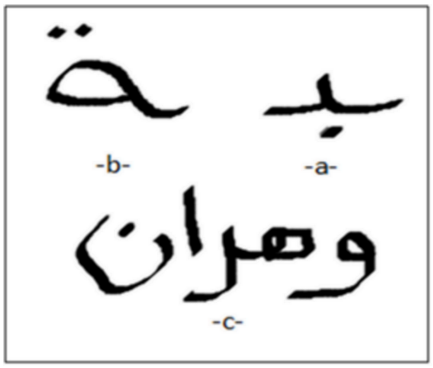

Figure 1. Online Arabic handwriting: (a) the "Bah" character in the middle of a word; (b) the "Tah" character at the end of a word; (c) the word "wahran" composed of 4 pseudo-words.

In this paper, the use of the on-line aspect decreases some processing difficulties because the diacritical points are considered as a signal, which complements the character body. Hence the fusion of diacritical points is not a real problem. However, the segmentation in pseudo-words or characters is not performed because the word is taken in its entirety, therefore, the overlaps are ignored. Nevertheless, these features increase the variability intra or inter-writer.

Among the approaches presented in the literature for a handwriting recognition, the neural networks demonstrate a great discriminating power and a very good capacity to construct class separators in multidimensional spaces. That being said, models based on hidden Markov models (HMM) [2] use a parametric approach to model sequences of observations. These sequences are generated by stochastic processes (e.g., handwriting manuscript), which are more visible in the word recognition process. The HMM has a strong

expressive power to model the distribution of observations for any class. For the character case, the neural networks are much more adapted because it is the global shape that dominates. These networks present the advantage to be compatible with the two dimensional pictorial nature of the writing.

1.1. Related work

The handwriting is a complicated task, due to the variability of handwriting styles. Indeed, the shape of the handwriting character differs not only from one writer to another, but also for a single writer according to: its position in the word, the neighbouring letters, the writer's health (motor disabilities, Parkinson ...), the psychological status (depression, sadness ...). In addition, it is also possible to have the same trace for two different meanings depending on the context. This represents another source of ambiguity. The aforementioned properties make the handwriting recognition more challenging and the complicated patterns recognition a problem.

The online handwriting recognition is not a recent subject. It goes back to around 30 years ago. However, the emergence of more sophisticated devices, like smartphones and tablets, has stimulated more research works in this field. The online recognition problem is characterized by three fundamental properties: line temporal chaining, the dynamic of the trace (speed, acceleration, pressure on the pen) and the trace skeleton (ignore the thickness of line).

In [3], Liu et al. presented the state of the art of online Chinese handwriting recognition. In [4], Kessentiniet al. have developed a unified approach for Arabic and Latin alphabetic scripts recognition.

However, few works were interested in the online Arabic writing recognition. In fact, its cursive nature triggers numerous problems in the automatic recognition system. These problems are bound to the diversity and the complexity of the handwriting signal. The unavailability of databases with online character also represents another problem for this field of research. Consequently, most of the works focused on the offline aspect, such as [5] where Margner et al. present a development strategy for offline Arabic manuscript recognition systems. More recent works have been presented in [6], which propose a deep architecture for the Arabic manuscript recognition. Nevertheless, several works treat the online case, like Mezkhani et al. They have used a Bayesian classification [7] and a Kohonen network [8] for the online Arabic character recognition. In [9], Biadsy used the hidden Marcov model for the Arabic manuscript recognition. And more recently, S.Benbakreti and A.Boukelif proposed a TDNN "time delay neural network" based recognition system of the online isolated Arabic characters [10].

In this work, we present a design of an Arabic handwriting recognition system; we propose a classification approach based on neural networks. Figure 2 illustrates the architecture of the proposed system.

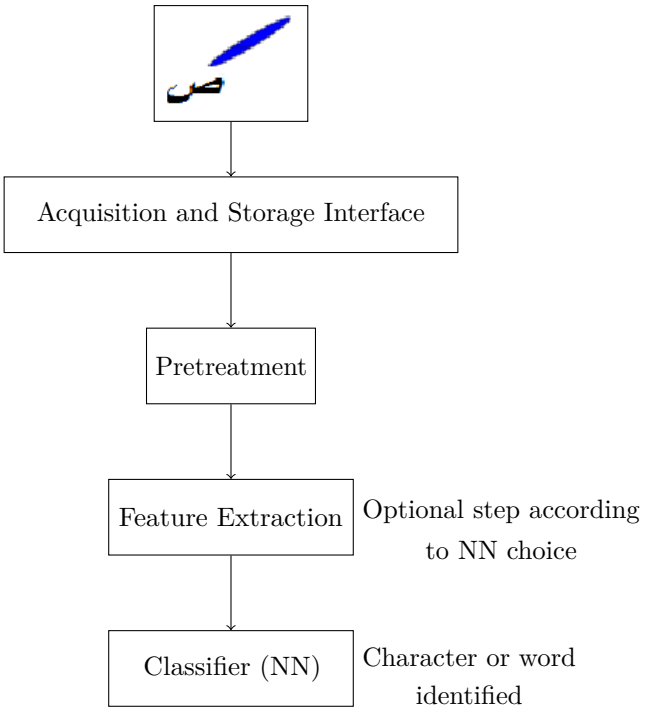

Figure 2. Architecture of the recognition system.

2. Processing and features extraction

The preprocessing stage described in the previous recognition system used for the character size normalization and trace sampling in a fixed number equidistant points fashion. The next stage extracts features from the previously sampled trace, which prepares the input for the neural network classifier. After the training phase, the class of the character is supplied at the output layer. This procedure is respected by almost all the neural networks. However, the deep neural networks do not require any features extraction phase. The task is done implicitly in the network.

The task of the character recognition is difficult, because of the big variability inherent to the handwriting style and the variation of the character position in the word (see Figure 3).

For most of the Arabic letters, similarities are noticed between forms at the beginning/ middle on the one hand and the end/ isolated on the other. The presence of a ligature with a previous/next letter does not significantly modify the form of the letter (no more than the case of Latin cursive handwriting). Arabic ligatures occur where two letters are written one upon the other. Furthermore, the Arabic writing is rich in diacritic signs (or secondary signs) such as vowels, points (dots), chaddah, maddah, hamzah, etc. In our work, we define a diacritical mark as a secondary component of a letter, which may complete it or even modify the whole sense of the word. But the notion of vowel, such as el damah, el kasrah, el fathah and el soukoun are not considered.

N°	Character name	Position in the word			
		End	Middle	begining	Isolated
1	Alef	أ			أ
2	Ba	لعب	صبر	بحر	ب
3	Ta	كرة	ستر	تمر	ت
4	Tha	نعت	نثر	ثلج	ث
5	Jim	ثلج	خجل	جمل	ج
6	Ha	ملح	بحر	حلم	ح
7	kha	صراخ	نخل	خجل	خ
8	Dal	يد			د
9	Dhal	معاذ			ذ
10	Ra	مصر			ر
11	Zain	معز			ز
12	Sin	فرس	يسر	سهل	س
13	Chin	نعش	نشر	شرق	ش
14	Sad	مص	مصر	صراخ	ص

N°	Character name	Position in the word			
		End	Middle	begining	Isolated
15	Dhad	مرض	مضلة	ضمير	ض
16	Ttah	تسليط	مطر	طفل	ط
17	Thah	حظ	نظر	ظهر	ظ
18	Ain	متاع	معطف	عنب	ع
19	Ghain	صبغ	لغم	غمامة	غ
20	Fa	انف	طفل	فم	ف
21	Qaf	ازرق	مقص	قمر	ق
22	Kaf	حراك	مكيال	كم	ك
23	Lam	محل	ملعب	لهو	ل
24	Mim	فم	ضمير	ملعب	م
25	Noun	لبن	عنب	نذل	ن
26	Waw	هو	اسود	وهر	و
27	He	مياه	نهر	هرم	ه
28	Ya	هي	ويل	يسر	ي

FIGURE 3. Different forms of the Arabic character according to its position in the word.

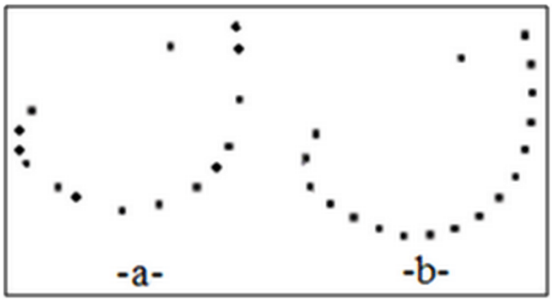

FIGURE 4. Result of preprocessing: (a) character Noun before preprocessing; (b) character Noun after preprocessing.

These specificities of the Arabic language make the recognition task more complicated, hence the necessity to do a preprocessing including the following steps:

- **A spatial sampling** that allows preserving the useful information of the signal and excluding the redundant information resulting from the repetition of points. Indeed, the duration of character formation as well as its shape can vary significantly from one writer to another and from an occurrence to another. The spatial sampling transforms a handwriting signal into a sequence of equally spaced points with coordinates $[X(n), Y(n)]$ (n is the point index) according to the length of the trace for a fixed number of points.
  - ▷ Definition of the line between pen up and pen down.
    N: number of sampling points
    Reservation of the memory space for the following vectors: X, Y, PenUp / Down
  - ▷ Calculation of the total length of the signal:
    If only one trait: Length = length + distance between points
    If several traits: Length = length + length between strokes.
  - ▷ Calculation of the sampling distance
    Dist-samp = Length/$(N-1)$
    Determination of $N-1$ points by interpolation
- **Character normalization and centring:** The character is recentred to (x_0, y_0) then normalized to the maximal size of the character. To obtain an invariant representation with respect to the translations and the spatial distortions.

A result sample of these two preprocessing steps is illustrated in Figure 4:

Once the preprocessing step is finished, in order to facilitate the classifier task, geometric information, such as the direction of movement and the curvature of the trajectory, are extracted from this sequence of points. We then obtain a vector sequence of 7 characteristics, which are mentioned in the following:

- The x coordinates
- The y coordinates

Algorithm 1

$xmax \leftarrow x(0)$
$xmin \leftarrow x(0)$
$ymax \leftarrow y(0)$
$ymin \leftarrow y(0)$
{Calculation of the center of the character $[x0, y0]$}
$n \leftarrow 0$
while $n \leq N - 1$ **do**
  if $x(n) > xmax$ **then** $xmax \leftarrow x(n)$ **end if**
  if $x(n) < xmin$ **then** $xmin \leftarrow x(n)$ **end if**
  if $y(n) > ymax$ **then** $ymax \leftarrow y(n)$ **end if**
  if $y(n) < ymin$ **then** $ymin \leftarrow y(n)$ **end if**
end while
$x0 \leftarrow (xmin + xmax)/2$
$y0 \leftarrow (ymax + ymin)/2$
{*Delta* scaling factor calculation}
$Delta_y \leftarrow (ymax - ymin)$
$Delta_x \leftarrow (xmax - xmin)$
if $Delta_x < Delta_y$ **then**
  $Delta \leftarrow Delta_y$
else
  $Delta \leftarrow Delta_x$
end if
{Calculation of the new coordinates $x[n][1]$ and $x[n][2]$: $x[n][1]$ the x coordinates $x[n][2]$ the y coordiantes. Scanning of (X and Y) points}
$x[n][1] \leftarrow (X[n] - x0)/Delta$
$x[n][2] \leftarrow (Y[n] - y0)/Delta$
$x[n][7] \leftarrow penUpDown[n]$
{Calculation of the direction $x[n][3]$ and $x[n][4]$: cosine directions Scanning points Calculation of the length of the dS string}
$dx \leftarrow X[i + 1] * X[i - 1]$
$dy \leftarrow Y[i + 1] * Y[i - 1]$
$dS \leftarrow \text{sqrt}(dx * dx + dy * dy)$
if $dS = 0$ **then**
  $x[i][3] \leftarrow 0$
  $x[i][4] \leftarrow 0$
else
  $x[i][3] \leftarrow arc_x/dS$
  $x[i][4] \leftarrow arc_y/dS$
end if
{Special points: initial and last point}
Initial point $\leftarrow$ *next point*
Last point $\leftarrow$ *previous point*
{Curvature calculation $x[n][5]$ and $x[n][6]$: cosine directions Scanning points}
$x[i][5] \leftarrow x[i+1][3] * x[i-1][3] + x[i+1][4] * x[i-1][4]$
$x[i][6] \leftarrow x[i+1][3] * x[i-1][4] + x[i+1][4] * x[i-1][3]$
{Special points: initial and last point}
Initialpoint $\leftarrow$ *nextpoint*
Lastpoint $\leftarrow$ *previouspoint*

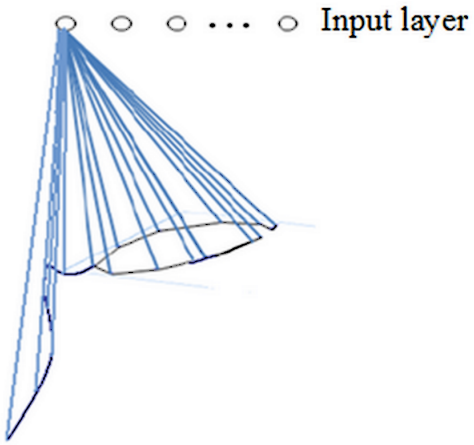

FIGURE 5. MLP type connexion -++ complete field of view.

- The cosine directions of the movement direction ($\cos\theta$ et $\sin\theta$)
- The cosine directions of the trajectory curvature ($\cos\Phi$ et $\sin\Phi$)
- The pen up/down.

This step is summarized in Algorithm 1.

Once the features extraction is realized, we can proceed to the classification.

3. NEURAL NETWORK CLASSIFICATION

We have used a set of neuronal classifiers, which gave satisfactory results, the proposed neural architectures are as follows.

3.1. MULTI-LAYER PERCEPTRON (MLP)

We have used the MLP with error back-propagation that has one hidden layer only [11, 12]. The seven features extracted from the Arabic characters (characters in their isolated forms, at the beginning, middle and end), generated by the previous module, constitutes the inputs of the network. The hidden layer includes 80 neurons. The classes to be discriminated are 28 characters of the Arabic alphabet (or 48 words), hence the choice of 28 (48) neurons for the output layer. The unipolar sigmoid was used as a neuron activation function. Figure 5 shows how the MLP processes the letter Mim.

However, the training of the lowest layer (the closest to the output layer) is less efficient in a deep MLP [13, 14]. It seems that the update of the parameters is less and less relevant as we propagate in the lowest layers. Because of that, we use one hidden layer for our MLP.

3.2. TIME DELAY NEURAL NETWORK (TDNN)

The used TDNN consists of three layers: input, hidden and output. Furthermore, every layer possesses two directions: a direction of features and a temporal

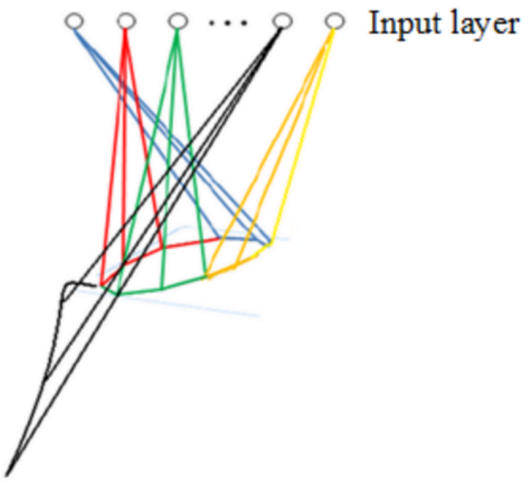

FIGURE 6. Convolution network connexion — restricted field of view.

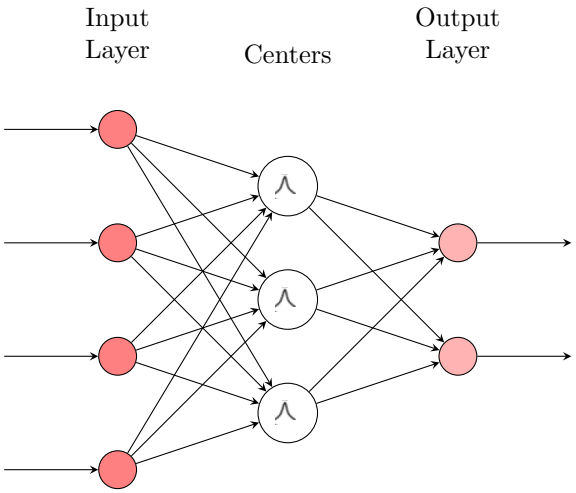

FIGURE 7. The RBF network architecture.

direction. The TDNN is distinguished from a multilayer perceptron by the fact that it takes into account a time notion. Consequently, it processes all the neurons of the input layer at the same time (see Figure 6), before making a temporal scanning. It is the notion of the temporal window.

The TDNN has three basic principles: temporal window, shared weights, and delays:

- The temporal window: the basic idea states that every neuron of the $L+1$ layer is only connected to one subset of neurons from the L layer. After an experiment study, we have fixed the size of this window to seven neurons.
- The shared weights: this notion allows reducing the number of network parameters, which positively influences the network generalization capacity. A window with a given characteristic will have the same weights according to the temporal direction. It also allows progressively extracting the differences when scanning the signal.
- The delays: in addition to the previous two constraints, we introduce delays between two successive windows for a given layer. The first delay is three neurons and the second one is four. For the shared weights constraint, the same neuron is duplicated in the time direction (the same duplicated weight matrix) to detect the presence or the absence of the same feature on various points in the signal. By using several neurons in every temporal position, the network of neuron detects the different features: the outputs of the different neurons of one layer produces new features vector for the next layer and so on. Thus, the temporal component of the original signal is progressively eliminated and transformed into a feature by upper layers. To compensate this loss of information, we increase the number of neurons in the feature direction.

3.3. Radial Basis Function network (RBF)

The RBF uses radial functions rather than sigmoid (e.g., MLP) to build a local decision function centred on a subset from the inputs [15]. The summation of all local functions represents the global decision function [16] to solve the local minima problem. The used RBF network [17, 18] consists of 3 layers (Figure 7). Every hidden node applies a kernel function to the input data, then the output layer performs a weighted sum of these kernel functions. Every node is characterized by two parameters, which are the width and the centre of the radial function. If the input data of a node are close to the centre (estimated using the Euclidian distance), the output value will be high. In this work, we have used a Gaussian kernel function.

The complete configuration of the network is achieved after determining the centre and the width associated to each node as well as the weight of the connections between the hidden layer and the output layer, which contains 28 Arabic characters or the 48 words representing the cities of Algeria. In this work, we have experimented with several variants of RBF networks, a brief description of each one of them is given in the following paragraphs:

3.3.1. Radial basis function networks exact (RBFNE)

The design of an RBFNE can be done using a function that takes as input : the matrices of input vectors P, the target vectors T and the spread factor of the radial basis layer, then it sends back a network with weights and bias values such that the output is exactly T when the input is P. The same function creates many radial basis neurons that input vectors in P. So, we have a radial basis neuron layer in which every neuron acts as a detector for a different input vector. Then, the only parameter that RBFNE has is the spread of the radial basis functions of the first layer. In this work, the value of the spread factor is 0.3.

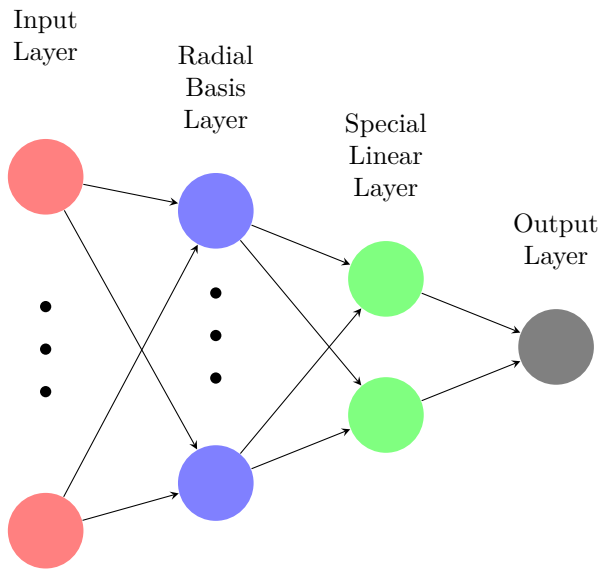

FIGURE 8. The GRNN network architecture.

3.3.2. RADIAL BASIS FUNCTION (RBF)

The second type iteratively creates a radial based neuron in each instant. The difference is that the RBF creates only one neuron at a time. In every iteration, the input vector, which will succeed to reduce most of the network error, is used to create a radial based neuron. The error of the new network is checked. If it is sufficiently small, the network creation is ended, otherwise, a following neuron is added. This procedure is repeated until the targeted mean squared error is reached ($10e^{-6}$), or the maximum number of neurons is reached (300). The number of neurons added between every evaluation is fixed, after several experiences, to 25.

3.3.3. GENERALIZED REGRESSION NEURAL NETWORKS (GRNN)

This variant is another type of the NN that was proposed by Speckt [19]. This regression based network estimates the expected mean value of the output variable using Bayesian technique. Any continuous function approximated to a linear combination of Gaussian functions. The objective is to make a regression, i.e., to build a good approximation of a function, which is known by a limited number of experimental points. This network can be used to solve the classification problem. For each input, the first layer calculates the distances between the input vector and the weight vector and then it produces a vector, which is multiplied by the bias. The network GRNN diagram is described in Figure 8:

The other GRNN specific layer is called a special linear layer and consist of two subparts: Numerator part performs a summation of the multiplication of training output data and activation function, the Denominator part is the summation of all activation function. This layer feeds both the Numerator & Denominator to the next output layer.

3.3.4. PROBABILISTIC NEURAL NETWORKS (PNN)

The probabilistic neural network, introduced by Donald Specht in 1988, is a feedforward network with three layers used for the classification of the data [20]. Contrary to the other neural networks, which are based on the backpropagation method, the PNN is based on the Bayes decision strategy and the probability density estimation. The PNN uses radial basis spherical Gaussian functions centred in each training vector. The membership probability of a vector in a given class is expressed as follows [21, 22]:

$$f_i(x) = \frac{1}{2\pi^{p/2}\sigma^p M_i} \sum_{j=1}^{M} e^{\frac{-(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}}, \qquad (1)$$

where i is the number of classes (28 or 48 in our case), j is the number of the forms to be recognized, M_i is the number of the training vector of the ith class, x is a test vector, x_{ij} is the jth training vector of the ith class, p is the dimension of the vector x, is the standard deviation, and $f_i(x)$ is the summation of the Gaussian spherical multi-variable functions, centred on each training vectors x_{ij} used for the evaluation of the probability density function of the ith class. The classification decisions are taken according to the decision of Bayes rule [22]:

$$d(x) = C_i \text{ if } f_i(x) > f_k(x) \text{ for } k \neq i, \qquad (2)$$

where C_i is the ith class. When an input is presented;

- The first layer calculates the distances between the input vector and all the training vectors, and produces a vector with elements that indicate how this input vector is close to every training vector.
- The second layer adds these contributions for every inputs class to produce a probability vector at the network output. Finally, a transfer function at the output of the second layer selects the maximum of these probabilities, and assigns "1" to the corresponding class and "0" to the others.

3.4. DEEP LEARNING

The deep learning models are built in the same way as the previously described multilayer perceptron, except that the different middle layers are more numerous. But the main difference is the learning method, which distinguishes them from a classic MLP and which gave them a renewed interest since 2006. In fact, the limits and the drawbacks [23] of the classic architectures mentioned so far in this paper contributed to this interest. For these reasons, Bengio et al. [11] proposed a greedy learning algorithm based on a staking of auto-associators, which allows building the hidden layers one after the other.

In this paper, we have used the deep network model proposed by Hinton et al. [12] in 2006, which is based on staked restricted Boltzmann machines (RBMs), to construct the so called Deep Belief Networks. The topology and the learning method used in this model are presented below.

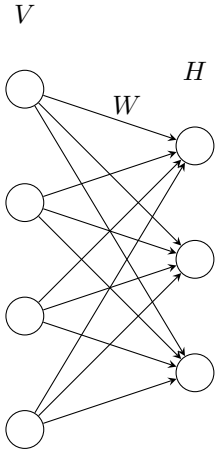

FIGURE 9. RBM: V is the visible layer, H is the hidden layer and W is the connection weight.

3.4.1. RESTRICTED BOLTZMANN MACHINE (RBM) AND CONTRASTIVE DIVERGENCE LEARNING (CD)

RBM is an undirected graph with two layers. The first layer is considered visible while the second is hidden. Each node represents a neuron. The neuron is active when its cost is equal to 1 and inactive if it is 0. The visible layer is connected to the hidden layer by weighted edges (Figure 9). The visible layer represents the observed data and the hidden layer represents the unknown elements associated with the data. Its size (fixed arbitrarily) allows the RBM to model more or less complex distributions.

Let us suppose that a bias unit, always active, is presented in the visible layer and the hidden layer. W is the weights matrix, where W_{ij} represents the weight of the link between the units v_j and h_i The RBM energy is given by:

$$Energy(v,h) = -h^T W v, \tag{3}$$

Each unit corresponds to a hypothesis to which we assign a binary value(1 or 0). The connections represent constraints for these hypotheses: if W_{ij} is negative, i and j should not be activated at the same time to reduce the RBM energy. But, if the weight W_{ij} is positive, then the simultaneous activation of both units reduces the energy. To summarize, the energy is a function of configuration, which is dependent on the constraints related to the weights. This energy function allows associating to an RBM, of weight W, a probability on the (v,h) configurations space:

$$P_w(v,h) = \frac{e^{-Energy(v,h)}}{Z}, \tag{4}$$

where Z is a partition function, it is the sum of all possible joined configurations. It is given by the following formula:

$$Z = \sum_{v,h} e^{-Energy(v,h)}. \tag{5}$$

The activation probability of the visible or hidden units are independent from each other. Let $sgm(t)$ be the sigmoid function:

$$sgm(t) = \frac{1}{1+e^{-t}}, \tag{6}$$

The conditional probabilities for an RBM are given by:

$$\forall i \le r,\ P_W(h_i \setminus v) = sgm\Big(\sum_j w_{ij} v_j\Big), \tag{7}$$

$$\forall j \le q,\ P_W(v_j \setminus h) = sgm\Big(\sum_i w_{ij} h_i\Big). \tag{8}$$

An RBM models the probability of an input v with the joined $P_W(v,h)$. We use the Gibbs sampling technique to make a random draw from the model to generate the configurations (v,h). It serves as reliable examples of inputs v.

This technique uses the Markov Chain Monte Carlo method, which consists in a repeated random draw from $P_W(h \setminus v)$ and $P_W(v \setminus h)$, such that the Markov Chain is guaranteed to be converged to the P_W distribution. In summary, this distribution is modelled by the RBM independently of the inputs, and according to it we can make a random draw with the Gibbs sampling method.

In the training part, we define P_{TR} as the probability distribution corresponding to the random draw of a sample v from a training database TR, then to the random draw of a hidden representation h, which is associated to v. P_{TR} is the objective distribution of training, it is fixed from a sample basis. Having defined P_W and P_{TR} , the training step will minimize the KullbackLeibler divergence between these two distributions in such way that the probability to draw a sample v from the examples approaches the probability to generate it from the training RBM.

It turns out that the KullbackLeibler divergence minimization has a very high treatment cost, thus we prefer to minimize an approximated criterion: Contrastive Divergence (CD). This technique was developed by Geoffrey Hinton in 2001.

The pioneering paper, written in 2002 [24], demonstrated an improvement of the handwritten digit recognition results (dataset MNIST) by using a CD algorithm to train RBM.

3.4.2. DEEP BELIEF NETWORK (DBN)

The restricted Boltzmann machines stacked in a generative way represents a particular type of a deep neural network, the reason why we detailed the RBM structure previously. The topology of such network is presented in Figure 10. The training of the stacked RBM is made layer by layer, in a greedy way. A first RBM is trained on the dataset of samples to minimize the Contrastive Divergence (CD). Then, each of the following RBM makes its training on the hidden representations of the previous RBM.

The training process includes two phases:

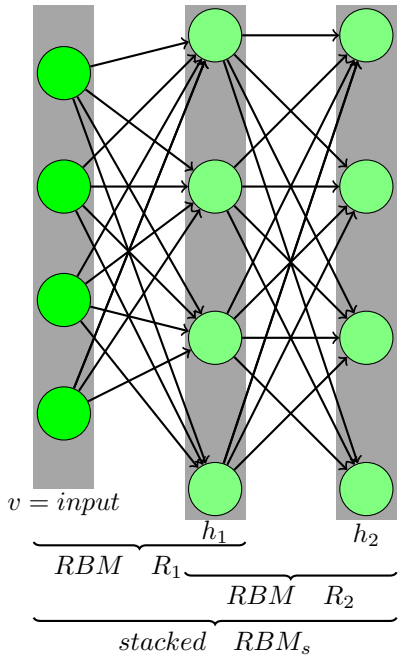

FIGURE 10. A deep network topology: stacked RBM.

(1.) **Unsupervised pre-training:** the DBN distinguishes itself from the other neural networks by the way of learning layer-by-layer, the idea is to train every layer as an RBM. The training begins from the lowest hidden layers (close to the visible input vector) and progresses to the output vector, so a DBN learns to probabilistically reconstruct its inputs. The layers then act as feature detectors. The DBN training algorithm for each RBM layer is presented as follows:

Step 1: build a multilayer network from the restricted Boltzmann machines (RBM), then train the layers by the Greedy-wise algorithm. Naturally, the input vector is the visible layer v.

Step 2: encode X as an RBM to produce a new sample v', by using the Gibbs sampling method.

Step 3: repeat the second step with $v \longleftarrow v'$ for each two successive layers until the two top layers of the network are reached.

(2.) **Supervising training:** the use of a supervised algorithm like back-propagation algorithm in MLP that helps to search the optimal parameters of the DBN network. Nevertheless, we use the weight W and the hidden bias b parameters to initialize this MLP.

In [13], several experiments were performed on different networks, trained for simple classification tasks (handwritten digit, geometrical forms). The conclusions of this reference paper are that training deep architectures was unsuccessful until the recent advent of algorithms based on an unsupervised pretraining. The concept of deep learning does not mean that the number of layers is high, it is rather the way of learning that is a different (layer by layer) method that has been applied in our work with the DBN network.

We deduce from these works that if the network contains enough number of neurons, the classification performances are better with an unsupervised training. Even on classical networks (not deep), the performances are improved. In our current work, we suggest the use of all the neural architectures described previously for the on-line Arabic handwriting recognition without constraint (letters and words). We also observe how the DBN manages to discriminate between the data without having a prior information and without making the feature extraction. In this classifier, the complexity resides in the choice of the hyper-parameters of the algorithm CD, this heuristic choice requires many experiments to find the optimal architecture, giving the best rates of classification.

Database	Number classes	Writers	Sample of training/ test
Class letters	28	20	1400/1400
Class word	48	20	2400/2400

TABLE 1. Description of the database.

4. Description of the Database

The objective of the architectures described previously is the classification and the recognition of Arabic characters/words coming from our database NOUN-DATABASE built from an on-line acquisition by means of a tablet of acquisition. This choice is motivated by the fact that most of the existing databases are off-line i.e.,containing images. In this work, we wanted to handle the on-line writing signal rather than the images. The Arabic alphabet consists of 28 letters of variable shapes depending on its position in the word. Our database contains 2800 Arabic characters written by 20 different writers, each writer inserts the alphabet 5 times (once for every position in the word: in its isolated form, at the beginning, middle, end, and another time in the choice of the writer). The constructed database includes 4800 words, inputted by the same writers (20). The words represent the 48 Algerian provinces (Wilaya). Each writer performs five insertions of the 48 Wilayas. The handwriting can be influenced by several parameters, such as the age, the sex and the state of the writer (each writer possesses an appropriate writing style). To preserve the variability of the writing signals, we took care to make the data acquisition (the samples database) by 20 people with different sexes and age categories. The database is summarized in Table 1.

5. Experiments and Results

In this section, we shall only consider the final results of our system of the Arabic handwritten character recognition, i.e. with all the parameter adjustments

NN	Arabs letters		Time for classification (sec)
	% correct	% error	
MLP	97.73	2.27	48.21
TDNN	49.46	50.54	2358.00
RBF	86.09	13.91	23.09
RBFe	75.39	24.61	8.68
GRNN	86.36	13.64	8.14
PNN	84.92	15.08	8.73

TABLE 2. Arabic characters recognition rates and classification time according the neural network.

NN	Arabs words		Time for classification (sec)
	% correct	% error	
MLP	85.31	14.69	27.01
TDNN	46.76	53.24	32017.00
RBF	81.85	18.15	37.57
RBFe	62.58	37.42	16.38
GRNN	78.44	21.56	16.74
PNN	78.44	21.56	17.77

TABLE 3. Arabic words recognition rates and classification time according the neural network.

made in the different neuronal approaches. We also proceeded to the division of the database in two parts: 50 % of samples formed the training (1400 characters and 2400 words) and the remaining 50 % for the test (1400 characters and 2400 words).

5.1. Neural Approach (classical methods)

Experiment 1: In the first experiment, we have used the classic architectures described previously for the on-line Arabic handwritten character recognition regardless of its position in the word, i.e., in its isolated form, at the beginning, middle and end. The best configurations of our system gave the results illustrated in the Table 2.

Discussion: We notice that the used classic architectures give more or less satisfactory results, having said that, the best classification results of the Arabic characters were obtained by using the multi-layer perceptron MLP with a 97.73 % success rate.

Experiment 2: In the second experiment, we are going to use the same previous architectures for the online Arabic words recognition, represented by 48 Wilayas of Algeria. The best configurations of our system gave the results illustrated in the Table 3.

Discussion: We notice that the word recognition results have considerably fallen in comparison with Table 2, this is due to the increase of the number of classes (which rises from 28 to 48). That can also be explained by the complexity of the data samples, indeed a word can consist of several pseudo-words which can contain one or several characters. However, the variability of the writing signal is richer in the case of the word writing. In addition, the reduction of the rate recognition can be explained by the main limitation of the classic architectures. If the architecture is too deep, the optimization of the parameters very often leads to a non-optimal local minima. The increase of the number of layers did not improve the results (some-times, it degraded them). For this reason, we had previously highlighted that the results mentioned are those of the ideal architecture (with the best parameters for each neural network, including the number of layers). As explained in [13, 14]), the surface of the error is not convex and is more irregular as the network is deep. Despite this, the MLP still gives the best results, which is 85.31 % with an acceptable time of classification of 27.01 seconds. Other observation is that the TDNN takes a long time for the classification, because of its complex architecture, which adds an additional dimension to the network, i.e. a temporal dimension in this particular case.

5.2. Deep Neural Approach

Experiment 3: To enhance that results, we will use a deep neuronal architecture of the type Deep Belief Network (DBN), which represents a stacking of restricted Boltzmann machines. It should be noted that the recognition architecture system is slightly modified because we have eliminated the "Features extraction" block (See Figure 2). This is due to the abstraction ability of the deep learning network. In fact, it allows automatically learning the different representations of data by an abstraction level from the raw data.

Phase 1: Pre-training (Un-supervising training without backpropagation,): In this phase, the algorithm utilizes a concatenation of the RBM layers to learn the distribution of the inputs data X (handwriting signal) without taking into consideration the Y labels. Each RBM layer (the two first layers in this case) has a more abstract representation than the precedent. Hinton et al. [12] proved that a greedy learning method for unsupervised training is effective. This method is also called contrastive divergence (CD). This phase, called sometimes pretraining, could be considered as an efficient initialisation of the DBN.

Phase 2: Fine-tuning (Supervising phase with back-propagation) The role of the second part of the DBN is to convert the abstract representation X into a Y labels that are used in the case of the supervised learning (the last RBM layer with 2000 units). The back-propagation algorithm is used to readjust the network parameters, and finally, the global optimal network could be obtained. The output layer has the number of classes (48 in this case). The training phase is considered as accomplished

Parameter	Value
Number of hidden layer	2
Units of 1st hidden layer	100
Units of 2nd hidden layer	100
Units of 3rd hidden layer	2000
Batch size	100
Number of RBM epochs	50

Table 4. DBN Parameters.

	Arabs words			Time for classification (sec)
NN	% error before BP	% error after BP	% correct	
DBN	33.75	2.92	97.08	252.21

Table 5. Arabic words recognition rates and classification time according the DBN network.

once the error between the output and desired one is stabilized. After that, we proceed to test another samples from the database (see Table 1). This phase, which is called fine-tuning, is more time consuming than the previous one. The parameters of the used DBN network for each layer (the pre-training and fine tuning phases) are mentioned in Table 4.
The above described DBN network architecture gave the results shown in Table 5.
Discussion: We can see that the results of the word classification are significantly improved with the use of the DBN. Indeed, in the previous table, the best rate of classification was 85.31 % with the MLP. It reaches a rate of 97.08 % with the DBN network with the RBM staked, giving a gain of 11.77 %. Also, we noted that the second step, the supervised part (after the back propagation), that is called "fine-tunning" is very important in terms of the error rate reduction (2.92 % of error rate) and turned out to be much more efficient than the pre-training step (33.75 % of error rate), because we do not start from an initial random solution.
Figure 11 represents a summary of the classification rates obtained for each used neural network.

6. Conclusion

In this paper, we have proposed a neuronal approach, which aims to develop a solution based on neural networks for the on-line Arabic manuscript recognition acquired dynamically. With the aim of performing the on-line recognition, we built our own database, containing 2800 characters and 4800 words. We divided our work in two parts; the first presents a classic neuronal approach using these different neural networks: MLP, TDNN, RBFNE, RBF, GRNN and PNN. Our experiments on the character recognition gave interesting results, such as 97.73 % success rate for

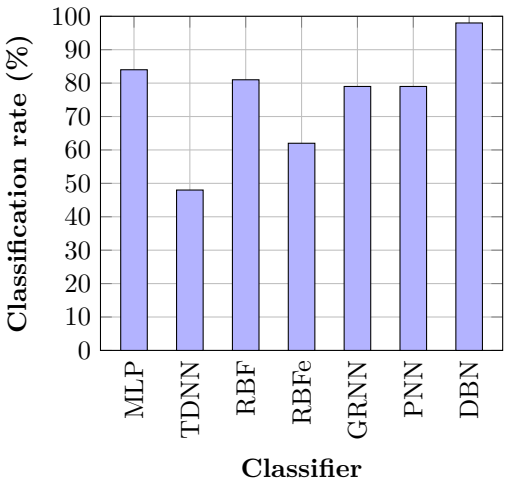

Figure 11. Comparison between the deep neuronal approach (DBN) and the classic neuronal approach for the Arabic word recognition.

the MLP and 86.36 % for the GRNN. Nevertheless, the application of the same architectures on the word recognition significantly deteriorates the results, which did not exceed 85.31 % for the MLP and 81.85 % for the RBF. This can be explained by the complexity of the input signal and the local minima problem. The second approach uses a deep learning, involving the DBN, which represents a stacking of restricted Boltzmann machines and which is characterized by a layer-by-layer learning method. This network significantly improves the performance for the words database and gives a classification rate of 97.08 %. In fact, the DBN has shown its ability to exerts an excellent performance for the classification tasks and dimension reduction. We conclude the superiority of the deep learning network compared to the classic neural networks. We also note that the various parameters used in all the previous neural architectures have allowed us to obtain the best classification rates.

References

[1] N. Ben Amara, A. Belaïd. Modélisation pseudo bidimensionnelle pour la reconnaissance de chaînes de caractères arabes imprimés. In *Colloque International Francophone sur l'ECrit et le Documernt - CIFEd'98*, pp. 131–141. Québec, Canada, 1998. Colloque avec actes et comité de lecture.

[2] R. Rabiner, L, H. Juang, B. An introduction to hidden markov models. pp. 4–16. IEEE ASSP Magazine, 1986.

[3] L. Cheng-Lin, J. Stefan, N. Masaki. Online recognition of chinese characters: The state-of-the-art. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(2):198–213, 2004. doi:10.1109/TPAMI.2004.1262182.

[4] K. Yousri, P. Thierry, B. H. AbdelMajid. A multi-lingual recognition system for arabic and latin handwriting. In *Proceedings of the 2009, 10th International Conference on Document Analysis and Recognition*, pp. 1196–1200. July 26-29, 2009. doi:10.1109/ICDAR.2009.55.

[5] M. Volker, E. A. Haikal. Databases and competitions: strategies to improve arabic recognition systems. In *Proceedings of the 2006 conference on Arabic and Chinese handwriting recognition*, pp. 82–103. College Park, MD, USA, September 27-28, 2006. DOI:10.1109/ICDAR.2009.55.

[6] E. Mohamed, T. Najiba, K. Monji. Optimization of dbn using regularization methods applied for recognizing arabic handwritten script. In *International Conference on Computational Science, ICCS*. 12-14 June 2017. DOI:10.1016/j.procs.2017.05.070.

[7] M. Neila, M. Amar, C. Mohamed. Bayes classification of online arabic characters by gibbs modeling of class conditional densities. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **30**(7):1121–1131, 2008. DOI:10.1109/TPAMI.2007.70753.

[8] M. Neila, C. Mohamed, M. Amar. Combination of pruned kohonen maps for on-line arabic characters recognition. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, p. 900. August 03-06, 2003. DOI:10.1109/ICDAR.2003.1227790.

[9] F. Biadsy, J. El-sana, N. Habash. Online arabic handwriting recognition using hidden markov models, 2006.

[10] S. Benbakreti, A. Boukelif. *Features Extraction and On-line Recognition of Isolated Arabic Characters*, pp. 481–500. 2018. DOI:10.1007/978-3-319-67056-0_23.

[11] Y. Bengio, P. Lamblin, V. Popovici, H. Larochelle. Greedy layer-wise training of deep networks. *in Advances in Neural Information Processing Systems 19,eds Schölkopf B, Platt J, Hoffman T, editors (Vancouver: MIT Press)* pp. 153–160, 2007.

[12] G. Hinton, S. Osindero, Yee-WhyeTeh. A fast learning algorithm for deep belief nets. *Neural Computation* **18**(7):1527–1554, 2006.

[13] E. Dumitru, M. Pierre-Antoine, B. Yoshua, et al. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *International Conference on artificial intelligence and statistics*, pp. 153–160. 2009.

[14] I. Guyon, P. Albrecht, I. Le Cun, al. Design of a neural network character recognizer for a touch terminal. *Pattern Recognition* **24**(2):105–119, 1991. DOI:10.1016/0031-3203(91)90081-F.

[15] P. Burrascano. Learning vector quantization for the probabilistic neural network. *IEEE Transactions on Neural Networks* **2**(4):458–461, 1991. DOI:10.1109/72.88165.

[16] K. Kim, D, H. Kim, D, K. Chang, SE, K. Chang, SA. Modified probabilistic neural network considering heterogeneous probabilistic density functions in the design of breakwater. *KSCE Journal of Civil Engineering* **11**(2):65–71, 2007. DOI:10.1007/BF02823849.

[17] J. D. Powell, M. Radial basis functions for multivariable interpolation: A review. In *Algorithms for Approximation*, pp. 143–167. Clarendon Press, Oxford, 1987.

[18] I. Park, W. Sandberg, I. Universal approximation using radial basis function networks. *Neural Computation* **3**(2):246–257, 1991. DOI:10.1162/neco.1991.3.2.246.

[19] D. Speckt. A generalized regression neural network. *IEEE Transactions on Neural Networks* **2**(6):568–576, 1991. DOI:10.1109/72.97934.

[20] F. Specht, D. Probabilistic neural networks. *Neural networks* **3**(1):109–118, 1990. DOI:10.1016/0893-6080(90)90049-Q.

[21] M. Berthold, J. Diamond. Constructive training of probabilistic neural networks. *Neurocomputing* **19**(1):167–183, 1998. DOI:10.1016/S0925-2312(97)00063-5.

[22] W. Tomasz, J. Jacek, M. Jacek. Probabilistic neural network for direction estimation. In *Proceedings of the Third Conference Neural Networks and their Applications and Summer School on Neural Networks Applications to Signal Processing, Kule, 14 X-18 X 97*, pp. 173–178. Eds R. Tadeusiewicz, L. Rutkowski, J. Chojcan. Częstochowa: Pol. Neural Netw. Soc. s., 1997.

[23] Y. Bengio, Y. LeCun. *Scaling learning algorithms towards AI. In Large-Scale Kernel Machines.* MIT Press, 2007.

[24] H. Geoffrey, E. Training products of experts by minimizing contrastive divergence. *Neural Computation* **14**(8):1771–1800, 2002. DOI:10.1162/089976602760128018.