Region Growing Based Segmentation Algorithm for Typewritten and Handwritten Text Recognition

Khalid Saeed, Faculty of computer Science, Bialystok Technical University, Bialystok, Poland
ksaeed@wi.pb.edu.pl
http://aragorn.pb.bialystok.pl/~zspinfo/

Majida Albakoor, Faculty of Science, Aleppo University, Aleppo, Syria
m-bakoor@scs-net.org

Abstract- This paper presents a new technique of high accuracy to recognize both typewritten and handwritten English and Arabic texts without thinning. After segmenting the text into lines (horizontal segmentation) and the lines into words, it separates the word into its letters. Separating a text line (row) into words and a word into letters is performed by using the region growing technique (implicit segmentation) on the basis of three essential lines in a text row. This saves time as there is no need to skeletonize or to physically isolate letters from the tested word whilst the input data involves only the basic information – the scanned text. The baseline is detected, the word contour is defined and the word is implicitly segmented into its letters according to a novel algorithm described in the paper. The extracted letter with its dots is used as one unit in the system of recognition. It is resized into a 9x9 matrix following bilinear interpolation after applying a lowpass filter to reduce aliasing. Then the elements are scaled to the interval [0,1]. The resulting array is considered as the input to the designed Neural Network. For typewritten texts, three types of Arabic letter fonts are used – Arial, Arabic Transparent and Simplified Arabic. The results showed an average recognition success rate of 93% for Arabic typewriting. This segmentation approach has also found its application in handwritten text where words are classified with a relatively high recognition rate for both Arabic and English languages. The experiments were performed in MATLAB and have shown promising results that can be a good base for further analysis and considerations of Arabic and other cursive language text recognition as well as English handwritten texts. For English handwritten classification, a success rate of about 80% in average was achieved while for Arabic handwritten text, the algorithm performance was successful in about 90%. The recent results have shown increasing success for both Arabic and English texts.

Keywords: Arabic Handwritten Text Recognition, Region Growing Segmentation, Implicit Segmentation, Contour detection, Horizontal Segmentation, Slant Correction

1. Introduction

Research on written texts and in particular written character recognition is as old as the computer itself, a proof of which is Ditten's work of 1955 [1]. However, its challenging nature and the continuously increasing demand for a fast cheap and more practical to implement recognition system has made it open for further research [2], [3]. OCR (Optical Character Recognition) systems have advanced far for typewritten texts of almost all typewritten languages including Arabic or other languages that use the Arabic alphabet (like Farsi, Kurdish and Urdu). However, the handwritten texts of these languages have received the least attention in this field [4-6]. This maybe is due to the lack or little interest in the field in addition to the specification of the Arabic language itself. The cursive nature of the Arabic text is the main obstacle to any Arabic OCR system [7]. In fact, the problem of cursive text classification exists even in English
language texts; most of the commercial classifying systems suffer from finding a solution to the problems of slant, slope and letter aliasing occurring in every handwritten text and even in many machine texts of cursive character [8], [9]. The cursive character causes problems resulting mainly from the concept of letter overlapping which is a steady feature of the handwritten texts taking place in all handwriting alphabets. Researchers have paid special attention to the problem of overlapping and developed many algorithms to segment the text into words and words into letters using varieties of techniques to overcome this problem, particularly in Arabic alphabet [10-12]. Many Arabic character recognition systems have been proposed [13]. Different approaches on both machine and handwritten texts have been presented and applied with efficient approaches to handwriting recognition [14-17]. In most of the text recognition systems, the character thinning to a one-pixel skeleton is required. Thinning methodologies can be found in detail in [18] and [19]. More methods and new approaches can also be found in [8] and [20]. However, thinning stage is bypassed in the work shown in this paper.

The major contribution of this paper is represented by implicitly segmenting (without visual cutting) the cursive words into their letters, and without thinning. Simply, the segmentation of the cursive word is impliedly achieved by only the idea of the baseline, the contour of the word and the growing region. This, noticeably, has saved time and the results are not worse than the known methods and rather better in some odd cases of poor handwriting.

The procedure of the new technique used in this paper starts with simple preprocessing stages. The image is converted into its binary equivalence by thresholding; the threshold is defined automatically when performing with MATLAB [21] in such a way that the handwritten text under consideration is completely separated from the background. The binarized image is also applied to a median-filter-supported 3x3 window [22], [23] for smoothing and noise elimination. In Matlab, the function ordfilt2 is used for enhancing the quality of the text image.

After the preprocessing stage, the text is separated into lines and the lines into words. The text segmentation into lines also implies the problem of overlapping but of horizontal type. Segmenting lines into words is also shown in this work in the section of vertical segmentation. The next step is to detect the baseline of the words and their contour by following the authors' previously implemented algorithms [8], [19], [24], [25]. Then the word is ready for further processing. For the sake of classification and recognition, the letter is considered together with its dots as one unit. It is then resized into an array of 81 elements with a bilinear interpolation by applying a lowpass filter. The size of the default filter is 3x3. The elements are rescaled into the interval [0, 1]. Finally, the array is entered as the input to the classifying Neural Network. The NN consists of one input layer of 81 neurons, one hidden layer of 200 neurons and an output layer of 28 neurons. Three types of Arabic typewritten-letter fonts with three different sizes are used; Arial, Arabic Transparent and Simplified Arabic. For the Arabic and English handwritten text classification, the following Darwin proverb is used:

"It's not the strongest that survives, nor the most intelligent; it's the one most adaptable to change."

In Arabic,
2. Text Processing for Region Detection: Lines and Words

The technique of histogram gives a good guidance to split the image into regions of similar type. However, the fundamental drawback of histogram-based region detection is that histograms provide no spatial information except the grey level distribution [26], [27]. From the other side, authors’ experiments on Arabic words have shown that the region growing technique allows treating the close pixels as ones of similar grey values [27]. On the basis of this idea and following the idea of the original methods of region growing [26-28] the authors have developed a modified region growing algorithm. This has enhanced the histogram to detect the region of words in a line and letters in a word without using the conventional segmentation. Thus, implicit segmentation is achieved. We will first consider the idea of baseline, upper and lower lines, and the region growing together with the algorithm that implies the new approach to the implicit segmentation of lines into words and words into their letters. An example on the typewritten word BAATIN باتن will accompany the explanation of the algorithm.

2.1 Word Detection

This starts with the detection of the baseline, the other two upper and lower lines and ends with contour detection. The word BAATIN باتن is considered for simplicity of algorithm explanation. To show how to extract the four letters from the word باتن, consider the following steps:

a. The main baseline is detected. The baseline plays an essential role in Arabic writing. Most letters are connected to each other at the baseline to form a zone of the highest pixel-concentration on the writing line. The baseline detection is performed by detecting the peak in the projection profile of the image (the horizontal histogram) of the written line. The baseline has the highest number of black pixels (Fig. 1).
b. There are two other lines to be detected, the upper and the lower lines. They efficiently help to find the positions of the letter-joining pixels on the baseline and they play an essential role in finding the word body. The position of the baseline is parameterized by the positions of these two lines, and vice versa. They are defined as the upper and the lower rows of largest number of pixels above and below the already evaluated baseline (Fig. 2). The baseline pixels in their major number are included between these two rows. Fig. 2 shows the plot of the horizontal projection profile of an Arabic text line. It illustrates the position of the upper and lower lines together with the baseline.

\[ \text{Fig. 1. The baseline of the text line computed on the basis of the compound word "بَاطِن" - Baatin} \]

(a)

(b)

\[ \text{Fig. 2. The plot of the horizontal histogram of the text line consisting of the word بَاطِن - Baatin is illustrating the positions of the three basic lines.} \]

(c. The contour of the word image is determined (Fig. 3). There are many methods available to thin the word to its contour. The method used here is given in detail in [8].

\[ \text{Fig. 3. The contour of the word "بَاطِن" -Baatin" together with the upper, base and lower lines} \]

2.2 Region growing method for letter detection

Here the basic idea of applying the region growing [26] to achieve implicit segmentation for letter detection is explained. Region Growing is an approach to image segmentation in which neighbouring pixels are examined and added to a region class (the word body here) if no edges are detected [27]. This process is repeated for each boundary pixel in the region. At the end of the section, a general region-merging algorithm is introduced together with its brief computer flowchart. Thus, the extracted word is first detected by defining the word shape without dots. The set of baseline points from the body of the word are extracted. These points, as stated before, exist only between the upper line and
lower line. The region growing concept is performed as shown in Fig. 4, which, in detail, illustrates the basic idea of the modified region growing algorithm as applied to the first part of the Arabic word بِاَمَه. It shows the extension of the connected neighbourhood pixels of the points and their next neighbours. More details on the basic region growing algorithms and their applications in Computer Graphics and Image Processing can be found in [26], [27] and [28]. After showing the method performance (Fig. 4) in MATLAB, the algorithm is then given.

**Fig. 4 (a)**

**Fig. 4(b)**

**Fig. 4(c) 1-10**

Fig. 4. The growing region. (a) shows the stages of the growing region for the subword "ایه". (b) and (c) show the performance stages of the modified growing region algorithm between step 1 and step 2 in (a). Notice the series of images 1-10 in (c) are given from left to right to show the step-by-step on-line scanning from right to left (Arabic text is read from right). in (a) shows an example of the first scanned feature points; they are the points between the upper and lower lines while shows a pixel \( x \) with its 8 neighbours and the way the region growing extends in (c).

**Algorithm**

This algorithm summarizes the detailed steps of the region growing in letter extraction.

1. After evaluating the word contour, first without dots, the vertical histogram of the resulted image \( \text{his}_{\text{vert}}(\text{word}) \) is computed.
2. The resulted word is scanned vertically from right to left.
3. If there is a pixel \( p(x_i, y_j) \) satisfying the following conditions, then the pixel \( p(x_i, y_j) \) is a joint pixel called \( p_{\text{joint}}(x_i, y_j) \), where \( i = d, \ldots, 1 \), and \( j = 1, \ldots, d_1 \) while \( d \) and \( d_1 \) are the width and height of the word image, respectively.
   - The position of \( p(x_i, y_j) \) \( \in \{\text{upper line, lower line}\} \) and \( p(x_i, y_j) = 1 \) & \( \text{his}_{\text{vert}}(x_i, y_j) = 2 \)
   - The position of \( P(x_{i-1}, y_j) \) \( \in \{\text{upper line, lower line}\} \) & \( p(x_{i-1}, y_j) = 1 \) & \( \text{his}_{\text{vert}}(x_{i-1}, y_j) = 2 \)
   - The position of \( P(x_{i-2}, y_j) \) \( \in \{\text{upper line, lower line}\} \) & \( p(x_{i-2}, y_j) = 1 \) & \( \text{his}_{\text{vert}}(x_{i-1}, y_j) = 2 \)
4. The point $p_{\text{joint}}(x_i, y_j)$ is not a joint pixel if the first pixel $p(x_k, y_{\text{min}})$ satisfies the following conditions, where $his_{\text{xor}}(x_i, y) > 2$ for $k = i - 1, i - 2, ..., 1$.

   ii. $p(x_i, y_{\text{min}}) = 1$
   
   iii. $p(x_{i-1}, y_{\text{min}}) = 1$
   
   iv. $y_a < y_b$ where $y_a, y_b$ are y-coordinates of $p(x_k, y_{\text{min}})$ and $p(x_{k-1}, y_{\text{min}})$, respectively
   
   v. $his_{\text{xor}}(x_i - 2, y) = 0$

Consequently, the other pixels belonging to the same letter, are defined.

5. Else, the point $p_{\text{joint}}(x_i, y_j)$ is a joint pixel and the letter is extracted. The start of a letter is indicated by the column of the scanned start and the end of the letter is indicated by the column of the joint pixel. For higher efficiency, the removed dots are added to the extracted letter. This is done as follows:

   e. The column of the joint pixel $p_{\text{joint}}(x_i, y_j)$ is checked for dots in the stored dot array.
   
   f. If there are dots and their number is tenacious to the extracted letter then the body of dots is considered as a part of that letter.
   
   g. Else, the body of dots is considered as a part of the next letter.

6. Repeat the steps from (1-5) until the word is scanned completely.

3. Segmentation Using Region Growing Algorithm

After we have seen how the algorithm of region growing works and how the word is implicitly segmented into its letters, we will now show the application of the algorithm to handwritten texts. The growing region is applied to both word detection in a line and letter detection in a word. Thus, the concept of both horizontal and vertical overlapping is considered.

In the following section the horizontal segmentation is considered for both cases - with and without word overlapping - in Arabic and English handwritten texts.

3.1 Horizontal segmentation – text into lines

The horizontal histogram is first evaluated and used to extract the text lines (text line will be called a row to avoid confusion with the base, upper or lower lines of a text line) from the image. This is done by computing the maximum value $n$ in the horizontal histogram of the row (its peak). Experiments have shown that $n/2$ is the optimal number of baseline total pixels per row in a text. Figure 5 illustrates the histogram of the first row showing the position of the base-line on it.

![Fig. 5 The first text row and its histogram to show the way of selecting the baseline position.](image)

Then the row separation starts by considering the Arabic text from right to left downwards searching for the pixel line that contains $n/2$ pixels or more. This line is then
the baseline of the first text line (row); its total number of pixels is evaluated and used as
the essential parameter in the growing region algorithm. On the other hand, the number of
pixels that exist above this line in the first text line will be the basis of upper and lower
parts verification of the considered line-text. Figure 6a shows the Arabic text under
testing while Fig. 6b shows the text after line separation.

(a) Overlapping in Arabic text  (b) Line separation

**Fig. 6.** Arabic handwritten text showing the horizontal overlapping problem.

Notice the extreme case of how the letter part that belongs to the word of the
fifth row appears as a part of the fourth row. Such cases are really the most difficult ones
and sometimes need the human check. In fact, it is this and similar problems that cause
the success rate to decrease below the expectation of researchers.

The same procedure in Arabic text processing is applied to English texts (Fig. 7a), but the
scanning is now from left to right. The text looks like Fig. 7b after line separation.

(a) Overlapping in English text

(b) The text after row separation

**Fig. 7** English handwritten text showing the horizontal overlapping problem.

The word letters that belong to one row but share space with the body of another word in
the preceding or following rows will find their correct place and belonging. Accordingly,
the "double l" in the word *intelligent* (Fig. 7a), which actually belong to the second row
are then evaluated to be within the second line and not the first although they are part of
both of the first and the second lines.
To show the detailed procedure of line separation from each other consider the English text part in Fig. 8. The extracted text row is divided into eight parts and the lower line is computed for each part. There are two lower lines, one for parts 1-3 and 6-8, the other for parts 4-5.

(a)  
(b)  

**Fig. 8** Separating two lines in English text

According to the algorithm the text parts of height $d_1$ in Fig. 8a (parts 4-5) is first looked at as part of the word ‘strongest’. When the region growing algorithm is applied, it will test the word ‘strongest’ within the first row. When it finds the large number of pixels within $d_1$ but below the lower line of the row of the word ‘strongest’, it will automatically exclude the additional pixels from this row and will therefore change direction from the letter ‘g’ to the letter ‘e’ and so on. This means the lower line of parts 1-3 and 6-8 is the actual line between the rows 1 and 2.
The same is repeated with the next rows until the end of the page.

### 3.2 Vertical segmentation – words into letters

The vertical histogram is used to extract the words from the text line. Here also the idea of region growing is used. Before scanning for letters in a word, both the slope and the slant are corrected. This will be shown on Arabic text.

**Slope correction**
The row is divided into four parts. The base-line in the first quarter (the most right) is found and then the other words in the row will be corrected according to it. This is because we assume people start writing straight but after the first words they write with an inclination and hence the first word would definitely decide in a high percentage the actual horizontal level of writing. All the words in a row are then shifted up to have their base-line at the same level of that in the first word. Fig. 9 shows what actually takes place when correcting the base-line of a row.
(a) The baseline of each word in a row is evaluated

(b) The slope correction

Fig. 9 Slope problem: the text row (a) before correction, (b) after correction.

Slant Correction
After the word slope is corrected, it is tested for the need to any slant correction. Figure 10 shows the word to be rotated for slant correction. It is then rotated by an angle calculated by the inclination of the baseline from the horizontal level (Fig. 10). This is easy to implement.

(a)  

(b)

Fig. 10 Slant correction. The word in (a) is rotated to the horizontal level in (b).

The whole text row containing the part illustrated in Fig. 10 is shown in Fig. 11 after slant correction.

(a)  

(b)

Fig. 11 Slope correction

Word stretching
There are two letters س and ص in Arabic alphabet which appear as a word rather than a letter as they seem to be composed of fragments similar to letters in a word. However, the system usually recognizes such cases by considering the distances between their composing parts and their heights. Moreover, in many cases we use the stretching technique, which solves many problems. Figure 12 illustrates the idea of stretching a word. It simply depends on extending the silence region around the baseline. If in the resulting object there are dots above or under the object, it means there exists a letter that was difficult to extract before stretching. Otherwise, if no dots exist, then the heights belong to only one letter. On the other hand, after stretching, we can see that the word is easy to look at as a whole and with clarity to see its letters. The region growing algorithm can easily find that the object in Fig. 12b is a word of three letters (Fig. 12c).
technique will enlarge the area of pixel connection between letters in a word and will certainly increase the probability of getting the right place of segmentation.

(a) A word  

(b) The word in (a) after stretching

(c) The three letters of the word in (a)

Fig. 12 Stretching technique as a tool in word segmentation into letters.

4. Neural Network Based Classification

In previous works other classifiers were used [29], [30], for example the conventional point-to-point comparison (or the nearest neighbour) classifiers. When classifying with these methods of simple comparison, the word feature vector \( x_i \) is compared with the vectors \( y_i \) describing the word in the Base Set to search for the nearest neighbour of the tested word. We used 1-norm distance \( d \), the Manhattan distance, as shown in Eq. (1), for those calculations:

\[
d = \sum_{i=1}^{n} |x_i - y_i|
\]  

(1)

This allowed achieving around 90% correctly recognized words with Arabic type writing. In some cases, the method of Nearest Neighbours is quite inconvenient because it needs to maintain big database of already classified vectors, and in order to recognize a word all the vectors from the database are compared with the vector describing the tested word. In cases of huge databases it becomes a time-consuming task and hence inefficient.

The use of artificial neural networks avoided such inconvenience. In our research we use a simple classic feed-forward neural network, with one hidden layer, trained by the backpropagation method [30]. As a transfer function, we took the bipolar logistic sigmoid function (Eq. 2)

\[
F(x) = \frac{2}{1 + e^{-x}} - 1
\]  

(2)

Two sets of documents are considered in this work. They are written in Arial, Arabic Transparent and simplified Arabic fonts of sizes 10, 12, 14 and 16. The first set is taken
of only size 14 for all fonts to be used in the training and teaching processes. The second set is taken of sizes 10, 12 and 16 of the considered fonts for the test process. All the documents are scanned at a resolution of 600 dots per inch. Each letter in the Arabic alphabet has four possible forms depending on their position in the word. They are isolated, beginning, middle or ending letters (see Table 1). Therefore, there are four cases in letter consideration and evaluation and hence, we use four NNs to classify the primary parts of the Arabic word. These NNs are trained to classify the Arabic letters as summarized in the results section below.

5. Results and Applications

In this section we show how to apply the neural networks to classify the Arabic word image after segmenting it into letters using the growing region technique. First, we consider the letters in their isolated forms (this is the case when the word structure involves letters in their isolated form – separated from each other like the word رازي in Table 1). Then, we show the classification results of the other three letter forms.

5.1 Arabic letters

Since the English typewritten letters are easy to recognize, we here consider only Arabic alphabet. The extracted letter image matrix is resized into a $9 \times 9$ matrix using the nearest neighbor interpolation. The resulted matrix is considered as an input to the NN. It is represented by one hidden layer of 200 neurons existing between an input layer of 81 neurons and output layer of 29 neurons due to the 28 Arabic basic letters plus the LamAlif (some references consider it as the 29th letter in the Arabic alphabet). Table 1 shows the output of the NN.

Table 1. Arabic alphabet in its 28 main letters with each given in its four different possible forms (from [8], I – Isolated, E – Ending, M – Middle, B – Beginning.)

This number of neurons in the hidden layer was found experimentally. We have experienced 100 and 200 neurons. The highest performance was at 200 neurons. Moreover, the number of iterations was 555 for 100 neurons, while it was only 338 for 200, with the best success rate. For example, the recognition rate of the letter jeem ج is 66.66% at 100 neurons while it is 91.66% at 200 neurons in the hidden layer. Table 2 shows how the rate of successful recognition increases when increasing the number of neurons from 100 to 200 in the hidden layer.

Table 2. The behaviour of NN classifiers for two different numbers of neurons in the hidden layer

The NN is trained on the first set depending on the default values as training parameters. The momentum coefficient is 0.25, learning rate is 0.05 and the sum square error is examined on 0.00000585. NN uses bipolar function as an activation function while the
NN weights and biases are generated randomly, and the range of input vector values is ranked between 0 and +1. The first set contains 87 letters used as examples through the training and learning processes while the second set contains 261 letters for the testing process. Figure 13 shows the plot of NN training.

![Fig. 13 The plot of NN training](image)

The output of the NN is set to 29. The NN is trained on 87 Arabic letters of size 14 of Arial, Arabic Transparent and Simplified Arabic fonts, and is recalled for 87 letters. The experiments held at these conditions gave a 100% recognition rate at training time of 3 [seconds] with 338 [epochs]. For Arial font at size 10 and 16, the rate of recognition was 100% but for size 12, the rate of recognition was 96.5%. For the Simplified Arabic font of size 10, the recognition rate was 96.5% while for sizes 12 and 16 it was 100%. For the Arabic Transparent font of sizes 10 and 16, the recognition rate was 100% but for size 12, however, the recognition rate dropped to 93.10%.

5.2 Other possible letter forms

The same discussion applies to the letters in other sizes for different positions. Table 3 shows the classification results for other considered letters of different fonts and sizes.

**Table 3. Classification results**

The same discussion applies to the hand written text in Arabic and English for different letter positions. Table 4 shows the classification results for both Arabic and English letters in a word after implicit segmentation.

**Table 4. Classification results**

6. DISCUSSION and CONCLUSIONS

In this paper a new method is exploited for automatic recognition of printed and handwritten Arabic and English words. The developed method proved to be a good word-segmenting tool. The examined word *Baatin* بَاطِن was chosen for demonstration because it resembles two words as it contains two subwords. The novelty of the algorithm lies in applying the idea of implicit segmentation using the region growing principle. The criterion is based on dividing the text into rows (text lines) and then finding the baseline of the row with two other lines (upper and lower). Together with the contour of the word, these lines are the base for examining the pixels of the word body for the letter-connecting points using the region growing algorithm. The resulting data array from the processing is entered to a Neural Network for classification. In Arabic type written text
Three types of Arabic letter fonts of three different sizes were used - Arial, Arabic Transparent and Simplified Arabic. The performance was by MATLAB. The results of the experiments proved high performance and high speed of the recognising system. They also showed the successful application of the classical NN in classifying the word letters using the optimal number of iterations for the given number of neurons per hidden layer. Other Neural Networks were also tested but the results were not better. This is because the more precise the stages preceding classification are, the better classification results are. Therefore, any reasonable classifying tool would be a possibility.

To increase the rate of success in handwritten recognition, the slant and slope of the text rows were corrected using easy-to-implement techniques. As a result, the recognition reached about 90% in the case of Arabic text and above 80% for the English text. The results, however, were so promising that the authors have decided to go further with this new approach. The current and future work is on handwriting with bilingual texts using larger database size with more training of the neural networks.

Acknowledgement

This work was supported by the Rector of Bialystok University of Technology (grant number W/WI/10/07).

The authors are indebted to the anonymous Referees for their constructive remarks and comments. In fact, it is these comments which changed the direction of research to cover both typewritten and handwritten texts written in both Arabic and English languages, the subject in which we have obtained promising results.

REFERENCES


[21] MATLAB, Version 6.1.0.450 (R12.1) on PCWIN


ليس البقاء للزُهْوَى
وللذكر ذُكْرًا
وأما لصاحب أكير إمكانية
وللهذين التعبير
والتنين لما هو جبر
دارون
It's not the strongest that survives, nor the most intelligent, it's the one most adaptable to change.

Darwin
"It's not the strongest that survives, nor the most intelligent, it's the one most adaptable to change."

Darwin
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>It's not the strongest that survives</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
It's not the strongest that survives,
This is the row baseline, drawn according to that of the first word.
سس البقاء للحقوه
ليس البقاء للقوى
ليس البقاء للزمن
Figure

لا Çalış
Table 1. Arabic alphabet in its 28 main letters with each given in its four different possible forms (from [8], I – Isolated, E – Ending, M – Middle, B – Beginning.)

<table>
<thead>
<tr>
<th>Sq.</th>
<th>Letter, Name</th>
<th>I</th>
<th>E</th>
<th>M</th>
<th>B</th>
<th>Sq.</th>
<th>Letter, Name</th>
<th>I</th>
<th>E</th>
<th>M</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A, Alif</td>
<td>ا</td>
<td>ا</td>
<td>ا</td>
<td>ا</td>
<td>15</td>
<td>Dh, Dhad</td>
<td>ض</td>
<td>ض</td>
<td>ض</td>
<td>ض</td>
</tr>
<tr>
<td>2</td>
<td>B, Baa</td>
<td>ب</td>
<td>ب</td>
<td>ب</td>
<td>ب</td>
<td>16</td>
<td>Tth, Ttaa</td>
<td>ط</td>
<td>ط</td>
<td>ط</td>
<td>ط</td>
</tr>
<tr>
<td>3</td>
<td>T, Taa</td>
<td>ت</td>
<td>ت</td>
<td>ت</td>
<td>ت</td>
<td>17</td>
<td>Zh, Zhaa</td>
<td>ظ</td>
<td>ظ</td>
<td>ظ</td>
<td>ظ</td>
</tr>
<tr>
<td>4</td>
<td>Th (ø), Thaa</td>
<td>ث</td>
<td>ث</td>
<td>ث</td>
<td>ث</td>
<td>18</td>
<td>Ea, Ain</td>
<td>ع</td>
<td>ع</td>
<td>ع</td>
<td>ع</td>
</tr>
<tr>
<td>5</td>
<td>J, Jeem</td>
<td>ج</td>
<td>ج</td>
<td>ج</td>
<td>ج</td>
<td>19</td>
<td>Gh, Ghain</td>
<td>غ</td>
<td>غ</td>
<td>غ</td>
<td>غ</td>
</tr>
<tr>
<td>6</td>
<td>Hh, Hhaa</td>
<td>ح</td>
<td>ح</td>
<td>ح</td>
<td>ح</td>
<td>20</td>
<td>F, Faa</td>
<td>ف</td>
<td>ف</td>
<td>ف</td>
<td>ف</td>
</tr>
<tr>
<td>7</td>
<td>Kh, Khaa</td>
<td>خ</td>
<td>خ</td>
<td>خ</td>
<td>خ</td>
<td>21</td>
<td>Qq, Qqaf</td>
<td>ق</td>
<td>ق</td>
<td>ق</td>
<td>ق</td>
</tr>
<tr>
<td>8</td>
<td>D, Dal</td>
<td>د</td>
<td>د</td>
<td>د</td>
<td>د</td>
<td>22</td>
<td>K, Kaf</td>
<td>ك</td>
<td>ك</td>
<td>ك</td>
<td>ك</td>
</tr>
<tr>
<td>9</td>
<td>Th, Thal</td>
<td>ذ</td>
<td>ذ</td>
<td>ذ</td>
<td>ذ</td>
<td>23</td>
<td>L, Lam</td>
<td>ل</td>
<td>ل</td>
<td>ل</td>
<td>ل</td>
</tr>
<tr>
<td>10</td>
<td>R, Raa</td>
<td>ر</td>
<td>ر</td>
<td>ر</td>
<td>ر</td>
<td>24</td>
<td>M, Meem</td>
<td>م</td>
<td>م</td>
<td>م</td>
<td>م</td>
</tr>
<tr>
<td>11</td>
<td>Z, Zay</td>
<td>ز</td>
<td>ز</td>
<td>ز</td>
<td>ز</td>
<td>25</td>
<td>N, Noon</td>
<td>ن</td>
<td>ن</td>
<td>ن</td>
<td>ن</td>
</tr>
<tr>
<td>12</td>
<td>S, Seen</td>
<td>س</td>
<td>س</td>
<td>س</td>
<td>س</td>
<td>26</td>
<td>H, Haa</td>
<td>ه</td>
<td>ه</td>
<td>ه</td>
<td>ه</td>
</tr>
<tr>
<td>13</td>
<td>Sh, Sheen</td>
<td>ش</td>
<td>ش</td>
<td>ش</td>
<td>ش</td>
<td>27</td>
<td>W, Waw</td>
<td>و</td>
<td>و</td>
<td>و</td>
<td>و</td>
</tr>
<tr>
<td>14</td>
<td>Ss, Ssad</td>
<td>ص</td>
<td>ص</td>
<td>ص</td>
<td>ص</td>
<td>28</td>
<td>Y, Yaa</td>
<td>ي</td>
<td>ي</td>
<td>ي</td>
<td>ي</td>
</tr>
</tbody>
</table>
**Table 2.** The behaviour of NN classifiers for two different numbers of neurons in the hidden layer

<table>
<thead>
<tr>
<th>Number of hidden layer neurons</th>
<th>Isolated</th>
<th>Beginning</th>
<th>Middle</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>98.26</td>
<td>82.79</td>
<td>86.36</td>
<td>91.86</td>
</tr>
<tr>
<td>200</td>
<td>98.85</td>
<td>86.59</td>
<td>90.10</td>
<td>93.02</td>
</tr>
</tbody>
</table>
Performance is $9.98574e-006$, Goal is $1e-005$
<table>
<thead>
<tr>
<th>Fonts and Sizes</th>
<th>Isolated all letters in</th>
<th>Beginning in</th>
<th>Middle in</th>
<th>End in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic Transparent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size 10</td>
<td>100</td>
<td>85.71</td>
<td>77.27</td>
<td>96.43</td>
</tr>
<tr>
<td>Size 12</td>
<td>93.10</td>
<td>89.66</td>
<td>93.10</td>
<td>86.21</td>
</tr>
<tr>
<td>Size 14</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Size 16</td>
<td>100</td>
<td>89.66</td>
<td>93.10</td>
<td>89.66</td>
</tr>
<tr>
<td>Simplified Arabic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size 10</td>
<td>96.5</td>
<td>71.43</td>
<td>77.29</td>
<td>85.71</td>
</tr>
<tr>
<td>Size 12</td>
<td>100</td>
<td>82.76</td>
<td>89.66</td>
<td>89.66</td>
</tr>
<tr>
<td>Size 14</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Size 16</td>
<td>100</td>
<td>93.10</td>
<td>93.10</td>
<td>100</td>
</tr>
<tr>
<td>Arial</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size 10</td>
<td>100</td>
<td>68.18</td>
<td>81.82</td>
<td>89.29</td>
</tr>
<tr>
<td>Size 12</td>
<td>96.55</td>
<td>79.31</td>
<td>93.10</td>
<td>96.55</td>
</tr>
<tr>
<td>Size 14</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Size 16</td>
<td>100</td>
<td>79.31</td>
<td>82.76</td>
<td>82.76</td>
</tr>
</tbody>
</table>
Table 4. Classification results

<table>
<thead>
<tr>
<th>Letter position (Arabic Handwritten)</th>
<th>Isolated all letters in</th>
<th>Beginning LS in</th>
<th>Middle LS in</th>
<th>End LS in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognition Rate of letters %</td>
<td>95.00</td>
<td>92.31</td>
<td>93.75</td>
<td>83.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Letter position (English Handwritten)</th>
<th>Isolated all letters in</th>
<th>Beginning CH in</th>
<th>Middle CH in</th>
<th>End CH in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognition Rate of letters %</td>
<td>83.33</td>
<td>82.35</td>
<td>76.47</td>
<td>80.00</td>
</tr>
</tbody>
</table>