Offline Persian Handwriting Recognition with CNN and RNN-CTC

Vahid Mohammadi Safarzadeh[1]
Research and Development Department
Takin Pardazesh Khayyam Company
Ahvaz, Iran
vahid.msafarzadeh@tpkai.com

Pourya Jafarzadeh
Research and Development Department
Takin Pardazesh Khayyam Company
Ahvaz, Iran
pouryajafarzadeh@tpkai.com

***Abstract*— Offline Persian handwriting recognition is a challenging task due to the cursive nature of the Persian scripts and similarity among the Persian alphabet letters. This paper presents a Persian handwritten word recognizer based on a sequence labeling method with deep convolutional neural networks (CNN) and recurrent neural networks (RNN). In addition, a connectionist temporal classification (CTC) loss function is utilized in order to eliminate the segmentation step required in conventional methods. The CNN layers are employed to extract the sequence of features from the word image. Altogether, the RNN layer with CTC function is used for labeling the input sequence. We showed that this combination is a robust recognizer for the Persian language as it was for other fields of application such as scene text recognition. The method is tested on the popular Persian and Arabic datasets including IFN/ENIT. It also is compared with novel methods and promising results have been obtained particularly in comparison with the conventional approaches including HMM and other machine learning-based methods.**

Keywords*— *Persian Handwriting, Word Recognition, Convolutional Neural Networks, Recurrent Neural Networks, LSTM, Connectionist Temporal Classification.

I. INTRODUCTION

Generally, the handwritten word recognition task is divided into two main groups: online and offline recognition. Online recognition depends on the trajectory of the pen and the coordinates of the pen's movement on the paper. But, offline recognition is done by analyzing the input image of the text. Online recognition is easier because we have both pen trajectory coordinates and the final image of the word. However, in the offline mode, we have only the image of the word [1]. Another difference between online and offline applications is in the shape of input data. In the online method, a 1-D vector is the input of the model that includes the features such as vertical position, writing direction, curvature, pen-up/pen-down, and Bézier Curves [2] [3]. But, in an offline method, the 2-D image of the word is the only accessible input [4].

Handwritten texts can also be divided into two cases: constrained and unconstrained. In a constrained handwritten text, the style of the character is limited to a special font. For example, the authors in [5] proposed a method for the Nastaliq (a traditional writing style in the Persian language) handwritten words recognition. On the other hand, in an unconstrained task, the style of the characters in the words is not limited. Consequently, recognition is harder in an unconstrained task and is still an open problem. In this paper, we propose a model for unconstrained handwriting recognition for the Persian language.

The Persian and Arabic scripts have cursive nature that means the characters in a word are connected to each other, unlike the English scripts that characters are disconnected in a word. So, conventional handwriting recognition methods divide their algorithms into two phases: segmentation and recognition. In the next paragraphs, we will describe these two steps.

Before segmenting a word, preprocessing operations like noise removal and slant correction should be applied to the image. For the segmentation, there exist various methods like finding minimum points of the word's contour [6]. In [5], they used this method for segmenting Persian Nastaliq handwritten words. In [7], the authors proposed a method for word segmentation using the points surrounding the baseline. Also, the authors used the same method in [8], in addition to applying a method based on the idea that the segmentation points are located at the end of a character and the beginning of its next character. In [9], an explicit segmentation is not applied, instead, they split a character into sub-characters and use the similarity between Arabic characters and different shapes of them based on their position in a word. In [10], they implemented the same method for Arabic word recognition. Although these methods have acceptable performances on word recognition, one disadvantage is that the segmentation is a time-consuming procedure. Furthermore, in unconstrained Persian written scripts there are some problems making the segmentation process even more challenging:

1- Sometimes characters overlap vertically;
2- Ascenders, descenders, and dots of each character don't have predefined forms and positions with respect to the main part of that character. For example, the ascender in the letter *Kaf* ("ﮐ") may be written on top of the previous character. It is also sometimes disconnected from the main part of *Kaf*. Fig.1 shows a word with ascenders, descender, and dots;
3- Many letters have different shapes in the words. For example, the letter Kh-e ("ﺧ") has different shapes at the end of the words, so the letter doesn't have a certain segmentation form;

[1] The authors contributed equally.

978-1-7281-5937-9/20/$31.00 ©2020 IEEE

4- The segmented parts of a letter may have similar shapes to other characters;
5- It is possible that after preprocessing and noise removal the main part of the input word be washed out and the segmentation algorithm will fail;
6- Due to the cursive nature and unconstrained writing style, the algorithm may produce extra segments in a character (over-segmentation). In addition, sometimes two consecutive characters may appear in one segment (under-segmentation).

Accordingly, we took into account a method for word recognition that doesn't need the segmentation step. So, we selected the connectionist temporal classification (CTC) [11] that make it possible to label the input without the need for segmentation. In section 2.3, the CTC method is described in detail. The proposed network converted the input image by convolution layers to a time sequence data and the recurrent layers with the CTC layer are used for sequence labeling.

The recognition phase starts by identifying the characters among segments. Conventionally, methods extract features from the segmented parts. Then, they identify characters using the features with the help of a statistical model such as Gaussian. Finally, the character sequence is recognized by a Hidden Markov Model (HMM) [12]. For instance, in [8], the feature vectors are created using discrete cosine transform coefficients. Then, a fully connected neural network with three layers is used. In [13], Bernoulli distribution is used with an HMM for word recognition. In [14], features are extracted by a hierarchical multilayer perceptron network; In [15], the features are the number of black pixels and the gradient of pixels; In [10], the features are extracted from a sliding window. In these last three works, they used a Gaussian model and an HMM for the purpose of character classification and word recognition, respectively. In [5], the features were Fourier descriptors and discrete and structural features. Then a mixture of Gaussian model is used to character classification, and a Variable Duration HMM [16] is applied for word recognition.

Although HHM has shown satisfactory performance in sequence labeling, it has a few drawbacks. One is that HMM assumes the probabilities of the observations are independent of each other, which is not true in handwritten texts. Alternatively, in the recurrent neural networks (RNN) the probabilities of the states are dependent on the adjacent states. Particularly, thanks to the bidirectional long short-term memory (BLSTM) block, the network access the contextual information in long-range in both forward and backward directions.

Another issue is that HMM is a generative method, while discriminative models have demonstrated better performances in labeling and classification problems. An RNN network trained by a discriminative function, like CTC, is a discriminative model and unlike generative methods, discriminative methods don't calculate the input distribution [17]. So, in the proposed model, we preferred BLSTM with CTC function to label the input Persian word sequence. Accordingly, we defined the Persian handwritten word recognition problem as a sequence labeling problem.

In recent years, some of the important problems in machine learning are proposed in the form of sequence labeling in online handwriting recognition [17] [18], offline handwriting recognition [19], scene text recognition [20], and speech recognition [21] [22] [23].

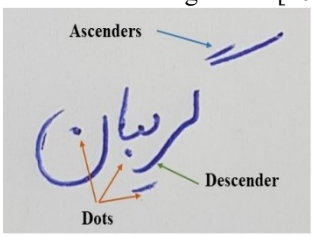

Fig.1. Ascenders, dots, and descender in the Persian word "گریبان",(Gariban, meaning collar)

A general definition for sequence labeling can be expressed as follows: the input of the algorithm is a sequence of data and the output is a sequence of discrete labels. In [24], a more formal definition is proposed: suppose that we have a training set S containing the data pairs $(\boldsymbol{x}, \boldsymbol{y})$, where x is the input vector and y is its corresponding target. The length of the target sequence $y = (y_1, y_2, \dots, y_M)$ is less than or equal to the length of the input sequence $x = (x_1, x_2, \dots, x_N)$: $|y| = M \leq |x| = N$.

Speech recognition and handwriting recognition are the most important application areas of sequence labeling. In speech recognition, the input is a sound signal and the output is the words corresponding to the input sound. On the other hand, in handwriting recognition, the input is handwritten words and the output is the characters.

In this paper, we introduce a novel Persian handwritten word recognition method consisting of a CNN to generate features sequence form the input word image, followed by stacked BLSTM layers with CTC function to label the input sequence by the Persian characters. We evaluated our model by various Persian and Arabic word, number and digit datasets including the well-known IFN/ENIT dataset [25]. We also compared our result with famous models in the literature. The proposed approach showed a comparable performance on all the datasets.

The rest of the paper is organized as follows: in section 2, we briefly present the concept and definition of our model's components including convolutional neural network, recurrent neural networks, LSTM, and the CTC loss function. In section 3, we describe the Persian and Arabic datasets precisely, and in section 4 the implementation details and experimental results on various datasets are reported. In section 5, the discussion regarding the functionality of the proposed model for the Persian words is discussed. Finally, in section 6, our conclusions and future works are provided.

II. METHOD

In this section, we describe the primary components of the proposed model.

A. Convolutional Neural Networks (CNN)

In the last decade with the progress of computational systems, abundance of the training data and the creation of regularization techniques such as batch normalization [26] and the dropout method [27], the application of convolutional neural networks [28] in machine learning projects have grown drastically. The input of the CNNs mostly are images, so they have many usages in image processing tasks. In [29], CNNs are used for image classification. In [30], they perform multi-object detection and

localization simultaneously on images. In [31], CNN is applied for digit recognition.

Generally, CNNs consist of multi-convolution layers connected sequentially to each other and followed by some fully connected layers. Each convolutional layer has inputs from the previous layer convolved with trained filters. The activation function of CNN layers is usually the $Relu$ function. Following the convolution, the pooling operation is applied to the output of the current layer to decrease the size of the data and reduce the overfitting of the network.

The top layers of the CNN are responsible for capturing primary shapes of an object such as edges and lines. The bottom layers detect more complex parts of an object like the tail, the face of a dog or the texts in an image. As a consequence, the top layers need a few epochs to converge. However, deeper layers converge after a considerable number of epochs. Moreover, if the input image is translated or rotated, it has significant effects on the first layers but has a much fewer effect on the last layers. Hence, the neural networks remain stable under most of the geometrical transformations [32]. Therefore, when the network needs more data for training, we can produce new data by applying translations on the input data as the work performed in [29].

The simplest way for improving CNN is increasing the size of the network's depth and width (number of filters in each layer). This is a useful solution when there exist a large amount of training data, but it has two disadvantages: firstly, by increasing the size of the network layers, the number of parameters increases and the networks becomes prone to overfitting. Secondly, by increasing the size of the network the computational cost increases dramatically [33].

B. Bidirectional Long-Short Term Memory

Recurrent Neural Network (RNN) is one of the most important types of neural networks that is proposed in order to deal with sequential data [34]. The idea is that the value of the data in the current time is dependent on the values of the previous times. Recently, many sequential labeling tasks such as: speech recognition, and online and offline handwriting recognition have used RNNs [35], [17], [19], [21], [18], [20], [22] . RNN's input mainly is a one-dimensional vector. Multi-dimensional data (e.g. images) also can be used as the input by transforming them into one-dimensional feature data.

Although RNNs have many applications in sequence labeling, they suffer from two important issues: firstly, the RNN can access the past context in the current time. Secondly, by increasing time steps, the backpropagation error blows up or vanishing. For case one the weights will oscillate and in the second case, the speed of updating weights is too slow [36]. The first problem is solved by using bidirectional neural networks [37] that encounters past and future information to produce the output of the current time. In fact, two RNNs move in both forward and backward directions through the sequence. The results of both forward and backward computations aggregate into the final network's output. The second problem is solved by utilizing long-short term memory (LSTM) [38]. LSTM is an important memory block for accessing data in long time steps and is similar to the memory chips in computers. LSTM has three primary multiplication gates: input, output and forget. They are similar to write, read and reset for a cell, respectively [35]. LSTM uses the multiplicative gates in order to store and access data over a long period of time, hence it eliminates the problem of vanishing gradient. The main part of the LSTM is the cell state, C_t, that conveys the main information through time. LSTM writes and clears the information in the cell state using the gates. Each gate consists of a sigmoid function and an element-wise multiplication operation. The output of the sigmoid function is between zero and one. Zero means nothing should be passed and one means the data should completely pass. Fig. 2 shows the architecture of the LSTM block. Now, we provide brief definitions of the gates.

Forget gate: the forget gate decides which of the information in the cell state should be saved and which of them should be discarded. The gate uses the output of the previous time step (h_{t-1}) and the input of the current time (x_t). The output is between 0 and 1; 0 means the data should be cleared and 1 means the data will be saved:

$$f_t = \sigma(W_f . x_t + W_f . h_{t-1} + b_f) \quad (1)$$

the values f_t are between 0 and 1. An element-wise multiplication of f_t by C_{t-1} determines which of the elements of the C_{t-1} must be saved and which of the elements must be cleared.

Input gate: this part has two phases, first, i_t decides which values are updated and g_t generates new values for the cell state:

$$i_t = \sigma(W_i . x_t + W_i . h_{t-1} + b_i) \quad (2)$$

$$g_t = tanh(W_g . x_t + W_g . h_{t-1} + b_g) \quad (3)$$

Hence, the new value of the cell state is the combination of the forget gate and the input gate. In fact, the forget gate clears the old data and saves the important data and the input gate writes on the cell sate:

$$C_t = f_t \otimes C_{t-1} + i_t \otimes g_t \quad (4)$$

Output gate: this gate controls which parts of the current cell sate (C_t) should be read and sent to the output.

$$o_t = \sigma(W_o . x_t + W_o . h_{t-1} + b_o) \quad (5)$$

$$y_t = h_t = o_t \otimes tanh\,(C_t) \quad (6)$$

In section 2 of [21], the architectures and equations of BRNN and LSTM are elucidated clearly. So, Bidirectional LSTM (BLSTM) is generated by combining BRNN and LSTM, that accesses long-range input in the past and future. In section 4.6 of [24], the equations of primary gates of BLSTM in the forward and backward steps are explained completely. This approach is also performed in [35], [22]. The results demonstrate that BLSTM is

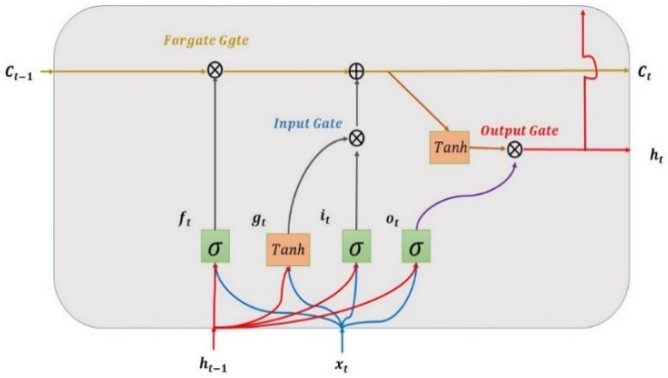

Fig. 2. An LSTM Block. An LSTM has three gates: Forget (yellow), Input (blue), and Output (red). Sigmoid functions (green boxes) play the main rules in the equations of the gates.

faster and more accurate than conventional RNN, BRNN, and LSTM.

C. Connectionist Temporal Classification (CTC):

Conventional methods of handwriting recognition require to segment the input word image into sub-words and characters. In these methods, the segmentation step plays a vital role and every segment is considered as an observation which probability of need to be calculated. But, we aimed to solve the problem without segmentation. In [11], the powerful connectionist temporal classification (CTC) method is proposed for sequence labeling without segmenting the input. In practice, CTC is a softmax layer following an RNN. Its outputs are the probabilities corresponding to all the label alignments to the input sequence with length T for all time steps.

In the CTC method, in addition to the alphabet labels (A), an additional label is defined as blank (denoted by "−"), that shows there is no label at a specific time in the output sequence. So, the new alphabet is: $A' = A \cup \{" - "\}$. In the output, the y_t^k is the activation of label $k \in A'$ at time t. Now, if we assume the probability at a time step independent of other time steps, the probability for assigning a sequence labeling (path) π to the input sequence x with the length T is:

$$p(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t \ , \quad \forall \pi \in A'^T \tag{7}$$

Moreover, a many-to-one function $\mathcal{F}: A'^T \mapsto A^{\leq T}$, that gets a sequence of length T composed of the alphabet A' as input, and generates an output sequence with a length less than or equal to T of alphabet A. More precisely, the function $\mathcal{F}$ removes blank labels and merges duplicate labels, For example:

$$\mathcal{F}(p - nn - n) = \mathcal{F}(-pp - -n - n) = pnn \tag{8}$$

So, the probability of a sequence labeling, $l \in A^{\leq T}$, is the summation of the probability of all paths that map to l by $\mathcal{F}$:

$$p(l|x) = \sum_{\pi \in \mathcal{F}^{-1}(l)} p(\pi|x) \tag{9}$$

where x is the input sequence.

Using the different paths for one specific labeling is the ability of CTC to use unsegmented data. This allows the network to predict the labels without knowing their position in the input sequence [24].

The CTC method has demonstrated its outperformance in the sequence labeling tasks such as speech recognition [21] and online and offline handwriting recognition [17], [19], [18]. For a more detailed explanation of CTC and the relative algorithms, we refer the readers to the section 7.3 and 7.5 of [24].

Nevertheless, for the tasks where the location of the labels should be determined in the input (the start and the end of characters in a word image), the CTC method is unsuitable [24].

As illustrated in Fig. 3, consider the image of a Persian number (Fig. 3.a) and its CTC output layer during the testing step (Fig. 3.b.) Each rectangle is the representative to the labeling that has the maximum value of the CTC function in the corresponding time step.

In Table I, the labels and their corresponding timesteps in Fig. 3.b is shown.

The output of the network is:

$$\mathcal{F}(1155588511774639) = 11585174639 \tag{10}$$

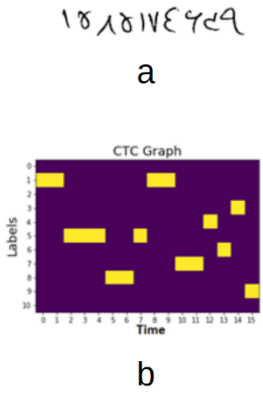

Fig. 3. A Persian number image "۱۵۸۵۱۷۴۶۳۹" (1585174639) and b. its corresponding CTC output

Table I. Output labels of CTC function in timesteps for input number "1585174639" (1585174639).

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Label	1	1	5	5	5	8	8	5	1	1	7	7	4	6	3	9

D. Network Architecture

The main structure of the proposed network is sketched in Fig. 4. It consists of deep CNN layers by stacking multi CNN layers on top of each other. The activation function of each convolutional layer is $Relu$. In order to avoid the vanishing or exploding gradient problem, the batch normalization technique is applied before $Relu$ function during the training phase. The output of the final CNN (conv10 in Fig. 4) layer is 512 feature vectors each of size 16. These vectors will be merged in order to generate a matrix of size 16×512 (16-timesteps). Then, this feature matrix will be used as the input sequence for the following BLSTM layers. We used two BLSTM layers with 512 LSTM blocks. The output of the first BLSTM layer is the input of the next. By assuming the same activation function $\mathcal{F}$ for all hidden BLSTM layers, the hidden sequence vector h_t^n of the nth layer is as follows [21]:

$$h_t^n = \mathcal{F}(W_{h^{n-1}h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n) \tag{11}$$

Furthermore, the second BLSTM has both backward ($\overleftarrow{h_t^2}$) and forward ($\overrightarrow{h_t^2}$) outputs with the size 16×512. Concatenating them ($\overleftarrow{h_t^2} + \overrightarrow{h_t^2}$) the output will become a 16×1024 matrix. The new matrix contains information of both backward and forward directions. Finally, the output of the deep BLSTM part will transform to a matrix of size $[16, number\_of\_labels + 1]$ (the +1 is for the blank label). This last matrix is the input of the CTC loss layer. In 16 timesteps, the probability of each label is calculated. Eventually, the most probable labeling is selected by the prefix search decoding method applied in [24].

III. Datasets

To evaluate the proposed method for the Persian handwritten word recognition, we applied the model on several standard Persian and Arabic datasets. The datasets include words and numbers. In this section, we describe the datasets in detail.

A. From resource [39]

In [39], a Persian dataset is proposed. The dataset includes sentences, words, numbers, dates, and characters. The dataset

contains written samples of 500 different male and female native writers. Among the writers 10% were left-handed and also the writers have different literature level with a various age range from 11 to 53. We used three sets: number strings, characters and words sets for evaluating our model. Below, we describe the format of these three sets.

1) Number Strings

The numbers set includes 39 different numbers with various lengths. The lengths of the numbers in the dataset are 2-digit, 3-digit, 4-digit, 5-digit, 6-digit, 7-digit, and 10-digit. Some of the 7-digit numbers are in the form of Persian dates. Therefore we didn't use the 7-digit numbers. In Table II, the statistics of the numbers. There are 17500 number images (excluding the 7-digit numbers), and we used 14000 (80%) of them for training and 3500 (20%) for the test.

Table II. Numeral string statistics in the dataset

Length of Number	*Total*	*Train*	*Test*
Two	5000	4000	1000
Three	3000	2400	600
Four	3000	2400	600
Five	2000	1600	400
Six	2500	2000	500
Ten	2000	1600	400
Two—Ten	17500	1400	3500

2) Alphabet Letters

There are 32 characters in the Persian language. In Persian and Arabic scripts, the form of a letter differs respect to its position in a word. For example, 25 characters in the Persian language have four different shapes (initial, medial, final and isolated) like "پ" (P-e) (پـ, ـپـ , ـپ, پ) , and 7 others have only 2 different shapes (final and isolated) like "و" ("ـو","و") (Vav). We test the system on the isolated characters set only, and their statics are mentioned in Table III.

Table III. Isolated letters statistics

Total	*Training set*	*Test set*
5000	34400	8600

3) Words

Handwritten words in the Persian language have cursive nature and the writing direction is from right to left. Furthermore, some characters have different writing styles in some positions, for instance, letter "ه" (H-e) in its medial form has two different shapes. In the dataset, the words are written by their different styles. The dataset includes 140 different words such as legal and natural person titles, commercial products, ordinal and cardinal numbers, name of the months in the Persian and Arabic calendars. In Table IV, the total number of words images is mentioned.

Table IV. Persian Words statistics

Total	*Training set*	*Test set*
70000	56000	14000

B. AHDBase

A large Arabic handwritten digits database (AHDBase) [40] is introduced. The dataset is composed of 70000 digits that 60000 of digits for training and 10000 for testing. The dataset has a format like MNIST [28] dataset and is written by 700 people by different ages and educational background. Arabic digits are similar to the Persian digits except at digit 6, Table V.

Table V. Arabic and Persian digits. The Arabic and Persian digits are similar except for digit 5.

English	0	1	2	3	4	5	6	7	8	9
Arabic	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
Persian	٠	١	٢	٣	۴	۵	۶	٧	٨	٩

C. Hoda

Hoda [41] is another large Persian digit dataset that is filled by B.Sc and high school students in Iran. The dataset has 60000 data for training and 20000 for the test (each digit class has 6000 sample for train and 2000 samples for testing) and the forms scanned at 200 dpi. The dataset is used in ICDAR 2009 Handwritten Farsi/Arabic Character Recognition Competition [42] in the digit recognition part.

D. IFN/ENIT

IFN/ENIT [25] is a famous Arabic cursive handwritten words dataset. The dataset consists of five sets: a, b, c, d, e, which contains 32492 images of handwritten 937 Tunisian town names. The dataset is completed by more than 1000 different writers. Moreover, two new sets f and s are generated for testing the ability of word recognizers. Set f is collected in Tunisia by the writers who had not participated in filling the first five sets and set s gathered from United Arab Emirates (UAE) by students in the University of Sharjah. In Table VI, the most common methods of splitting the dataset in order to evaluate a model are shown:

Table VI. Training and testing models for IFN/ENIT.

Train sets	*Test set*
a-b-c-d	e
a-b-c-d-e	f
a-b-c-d-e	s

IV. Experimental Results

To evaluate the proposed method for the Persian handwritten word recognition, we applied the model on standard Persian and Arabic datasets mentioned above. The implementation details and results are reported in the following sections. We used the model for evaluating words and numbers recognition in the Persian and Arabic languages.

A. Training and implementation details

The general structure of the network is shown in Fig 4. The size of the CTC layer (Fig. 3.b) depends on the dataset that is used for training and testing. The length of the labels-axis depends on the number of characters in the dataset, and the length of the time axis is greater than or equal to the length of the longest training label (y). For example, in the IFN/ENIT dataset there are 120 unique characters, so the length of the label-axis in CTC layer will be 121 (The Arabic characters in the dataset with different positions in the words, plus the blank character). The longest word in the dataset has 17 characters so we selected the time's length in the CTC layer 32. In practice, the length of the CTC's input is 32, and the output of the CTC (output of $\mathcal{F}$ function) is less than or equal to 32.

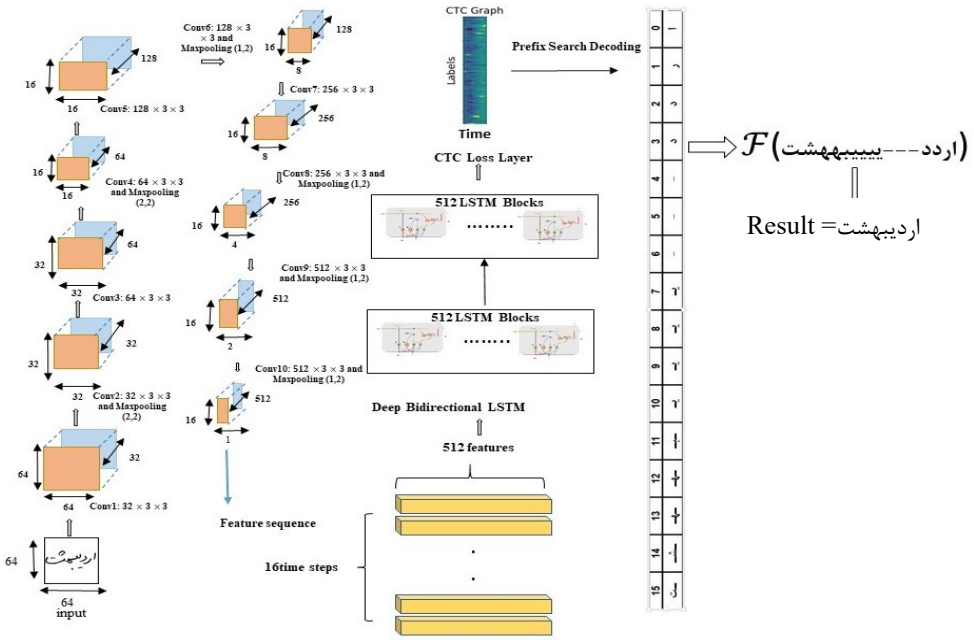

Fig. 4. The System Architecture. First, the input image (the image of the word "اردیبهشت", Ordibehesht, the name of the 2nd month of the Persian calendar) enters to the first CNN layer. Then, the output of the last CNN layer (features' sequence) enters into the BLSTM layer. Next, the output of the last BLSTM layer will be fed to the CTC. Then, the CTC function generates the most probable path in A' which is mapped to the final label by function F.

In the number string dataset, we have 10 unique characters so the length of the label-axis in CTC layer is 11 (we have 10 digits in the ground truth plus blank). The length of the labels in the training set is at most 10 (because of the number with length 10). In addition, for a similar reason as the previous example for the words, we set the time-axis in the CTC layer 16.

We used Tensorflow framework [43] to implement our network. For weights initialization, we used a truncated normal distribution method with zero mean and standard deviation of 0.1. In the training phase, the RMSprop [44] method is used with the learning rate 10^{-4} and decay rate of 0.9 and the training is stopped after 40 evaluation with no improvements. The training is carried out on a system with Intel Core i7-8550U CPU 1.80 GHz CPU, NVIDIA GeForce MX150 GPU, and 8.0 GB RAM.

One of the most important issues in pattern recognition problem is selecting the shape of the input data that it is valuable as the algorithm itself. Hence, before resizing the input image for entering the network we applied the following steps to the input image during training and testing phases:

1- Converting the grayscale image to a binary image by OTSU algorithm;
2- Removing small connected components (less than 5 pixels) considering them as noise;
3- Cropping the image around the input word image (removing white boundary around the word);
4- Inserting the word in the center of a square image.

In order to evaluate the performance of the network on the datasets, we used several measures. The most common measures in word recognition are character accuracy, Top1 and Top 5. Character accuracy is defined as:

$$accuracy = 100 * (1 - \frac{insertions+deletions+substitutions}{total\ length\ of\ all\ the\ words\ in\ test\ set}) \quad (12)$$

where the number of all insertions, deletions, and substitutions are summed up over all the words in the test sets.

Top 1 measurement is the rate of the number of times when the system's output is exactly equal to the expected answer. In word recognition, this is the rate of times that the first model's choice from the dictionary is exactly the expected word.

Top 5 is the rate of times that the expected word is among the first five model's choices from the dictionary.

In the evaluation of digit recognition, the measure is the percentage of correct recognized digits. Another important measure is the confusion matrix for analyzing missed digits. For number recognition, we used the same parameter as character accuracy for word recognition.

B. Evaluation

In this section, the evaluation of the method on various datasets is reported.

1) Results for resource [39]

The results of the test on the Persian dataset in [39] are presented in 3 sections: word recognition, number recognition, and character recognition:

a) Words

The dataset includes 56000 samples for training and 14000 for the test. In Persian language characters have different forms with respect to their position in a word. Consequently, for creating the labels of ground truth in the dataset we coded the words by symbols. In Table VII, word "دانشگاه" (Daneshgah meaning university) with its coded characters is expressed.

Table VII. We created the ground truth for the words in the dataset [39]. We coded the words by characters (like the second column). For example, letter Alef is coded by 'a' and '5' in isolated and final forms respectively.

Persian character	*Code in the ground truth*
د (isolated Dal)	D
ا (isolated Alef)	a
نـ (initial Noon)	6
ـشـ (medial Shin)	,
ـگـ (medial Gaf)	9
ـا (final Alef)	5
ه (Isolated He)	H
دانشگاه	Da6,95H

Table VIII shows the evaluation of the proposed method on word recognition in the dataset.

Table VIII. Word recognition accuracy on dataset [39]

Top1	*Top5*	*Character Accuracy*
98.87	99.68	99.35

b) Numbers

In Table IX the results of the evaluation of the model for number recognition in the dataset is expressed.

Table IX. Number recognition on dataset [39].

Length of number	*Digit Accuracy*
2	99.50
3	99.78
4	99.87
5	99.76
6	99.80
10	99.49
All Numbers	99.68

c) Characters

We evaluated character recognition for the isolated form of Persian characters in the dataset and the accuracy is 95.00. It is a robust result because some Persian characters are very similar in their isolated form. Some of them are just different in the number of dots such as "ب" (B-e) and "پ" (P-e), or they have similar main shapes like "و" (Vav) and "ر" (R-e).

2) AHDBase

In this experiment, we evaluated the model on AHDBase dataset for analyzing Arabic digits, (Table X).

In Table XI, the confusion matrix shows the statistics of false negatives and false positives of the test data:

Referring to the second row of Table V and analyzing the confusion matrix in Table XI, we can simply determine the Arabic digits that are similar to each other. For instance, digit 0 and 5 are so similar in the Arabic language. In Table XI, in the row of digit 0, we have 6 false-negatives (recognition 5 instead of 0). Similarly, in the row of digit 5, there are 9 false-negatives (recognition 0 instead of 5)

Table X. Recognition rate on digits recognition of dataset [40]. Our model outperforms the other model.

Method	*Recognition Rate*
ANN+SVM [40]	99.15
Ours	**99.43**

Table XI. Confusion matrix in Arabic digit recognition of dataset [40]. The red digits show the similarity between zero and five digits in the Arabic language.

		Recognized as									
	Digits	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
Input digit	***0***	**988**	5	0	1	0	*6*	0	0	0	0
	1	6	**991**	0	0	0	0	0	0	2	1
	2	2	0	**996**	0	1	1	0	0	0	0
	3	0	0	4	**996**	0	0	0	0	0	0
	4	0	0	5	0	**995**	0	0	0	0	0
	5	*9*	0	3	0	0	**986**	0	0	1	1
	6	0	2	0	0	0	0	**997**	1	0	0
	7	0	0	0	0	0	1	0	**999**	0	0
	8	1	0	1	0	0	0	0	0	**998**	0
	9	0	0	1	0	0	0	2	0	0	**997**

3) Hoda

Hoda is another Persian digit dataset that our model is applied to, and the results are reported in Table XII.

Table XII. Recognition rate on Persian digits recognition of Hoda dataset. Our method outperforms other methods.

Method	**Train**	**Test**	**Recognition Rate**
SVM [47]	60000	20000	99.02
SVM [48]	60000	20000	98.71
SVM [49]	60000	20000	98.55
Ours	60000	20000	**99.375**

Table XIII. Confusion matrix in Persian digit recognition of Hoda dataset. The red digits in the table show the similarity between digit 2 and 3.

		Recognized as									
	Digits	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
Input digit	***0***	1970	0	0	1	4	22	0	2	1	0
	1	0	1997	2	0	0	0	1	0	0	0
	2	0	1	1977	*16*	2	0	1	2	0	1
	3	0	0	*18*	1976	4	2	0	0	0	0
	4	0	0	5	4	1986	4	0	1	0	0
	5	3	0	0	0	1	1996	0	0	0	0
	6	0	1	2	0	0	4	1993	0	0	0
	7	0	0	1	0	1	1	0	1997	0	0
	8	0	0	0	0	0	1	0	0	1999	0
	9	2	7	0	0	0	4	3	0	0	1984

The confusion matrix demonstrates that many mistakes happen when people write 0 similarly to 5 in the Persian languages. It is obvious in the first row of Table XIII. Comparing the row of digits 2 and 3 it is clear that many people write 2 similarly to 3 (16 errors) and 3 similarly to 2 (18 errors).

4) IFN/ENIT

IFN/ENIT is one of the most famous Arabic datasets. The Arabic handwriting recognition competition (ICDAR) used this dataset for evaluating purpose. The first 17 systems' performances in Table XIV are taken from [45]. Due to the similarity between the Arabic and Persian handwritten styles we selected this dataset for testing our model.

In Arabic and Persian scripts there is a sign called Shaddah (Tashdid in Persian). It is a sign of emphasis on a letter in a word, Fig. 5. In IFN/ENIT, a letter with Shaddah has a different label from the times that it does not have Shaddah. So, to

Table XIV. Word recognition on IFN/ENIT dataset. Our model has a comparable performance with sophisticated methods.

Method	*abcd-e*		*abcde-f*		*abcde-s*	
	top1	*top5*	*top1*	*top5*	*top1*	*top5*
CACI-3	-	-	14.28	29.88	10.68	21.74
CACI-2	-	-	15.79	21.34	14.24	19.39
CEDAR	-	-	59.01	78.76	41.32	61.98
MITRE	-	-	61.70	81.61	49.91	70.50
UOB-ENST-1	-	-	79.10	87.69	64.97	78.39
PARIS V	-	-	80.18	91.09	64.38	78.12
ICRA	-	-	81.47	90.07	72.22	82.84
UOB-ENST-2	-	-	81.65	90.81	69.61	83.79
UOB-ENST-4	-	-	81.81	88.71	70.57	79.85
UOB-ENST-3	-	-	81.93	91.20	69.93	84.11
SIEMENS-1	-	-	82.77	92.37	68.09	81.70
MIE	-	-	83.34	91.67	68.40	80.93
SIEMENS-2	-	-	87.22	94.05	73.94	85.44
UOB-ENST	-	-	83.98	-	72.28	-
REGIM	-	-	57.93	-	72.28	-
Ai2A	-	-	89.42	-	76.66	-
MDLSTM [24]	-	-	93.37	-	81.06	-
MDLSTM [19]	-	-	91.43	96.12	87.83	88.00
RWTH-OCR	-	-	85.69	-	72.54	-
LITIS-MIRACL	-	-	82.09	-	74.51	-
LSTS	-	-	15.05	-	11.76	-
Segmentation-ANN [8]	90.37	-	-	-	-	-
MLP-GHMM [14]	92.70	-	90.90	-	81.10	-
(Bernoulli+HMM) [13]	-	-	92.20	-	84.62	-
MOG+HMM [15]	91.96	-	91.98	-	82.52	-
sub-character+HMM [10]	93.52	-	92.15	-	85.12	-
pixel-based and segment-based features + Gaussian [50]	93.90	-	93.20	-	88.50	-
MULTI-STAGE SUB-CORE-SHAPE HMMS [51]	93.9	-	93.32	-	86.52	-
CNN+N-grams [46]	97.07	-	96.76	-	94.09	-
Ours	**91.73**	**96.75**	**93.03**	**95.50**	**86.75**	**91.15**

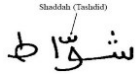

Fig. 5. An Arabic word with Shaddah (Tashdid)

simplify the labels and reduce the label size of the label-axis in the CTC layer, we take one label for the character in both states. Consequently, the size of the labels reduces significantly, so we will have 120 labels in the dataset.

In Table XIV, the comparison of our method with other methods is reported. Our results in Table XIV show that the proposed method has an acceptable performance on the IFN/ENIT dataset. The method [46] outperformed other methods. They have used nine convolutional layers in the networks and the output of the last layer of the CNN is followed by three parallel fully connected layers. The fully connected layers used to estimate the n-gram frequency profile contained in the word. Frequencies for unigrams, bigrams, and trigrams are calculated for the input word.

V. DISCUSSION

Most of the Persian characters have secondary stroke parts such as dots (like "پ") and ascenders (like "ک "). In some methods based on segmentation and hidden Markov models (HMM) the main stage of the algorithm is segmentation. The existence of dots, ascenders, and overlapping the characters with each other making the segmentation process hard. In the unconstrained Persian handwritten words, the secondary strokes have undetermined shape and positions. For example, dots or ascenders are eighter connected to the main part of the letter or farther from the main part. To deal with such conditions we used the CTC method in order to eliminate the requirement for the segmentation process.

Through CNN layers the network prepares the input feature vectors from the word input image to be used by BLSTM layers. RNNs assume that the input feature vectors in time series are dependent and create a nonlinear model between features. In the HMM-based handwritten word recognition models such [5], [15], and [10], the similarity probabilities of the observations are assumed independent of the adjacent states and just depends on the current states. In practice, in a handwritten word (e.g. in Persian) the elements of a sequence are influenced by the surrounding elements [17]. For instance, the character "ر" (R-e) has a different writing style when its previous character is "ب" (B-e) compared to the times when its previous character is "م" (Mim). Therefore, we found BLSTM applicable for accessing the contextual data in long-range and both directions. The proposed method had an acceptable result on the Persian and Arabic datasets.

In the evaluation of the network we found out that if there are at least 30 samples available for training a word, it will be very trustful in recognition of that word in the test phase.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed a model using the advantages of both CNN and RNN for the word recognition purpose. The method also benefits from CTC for eliminating the segmentation procedure. The efficiency of our method on several datasets are reported and our method showed comparable performance in comparison with other state-of-the-art works in the literature. For future work, we decide to use natural language processing (NLP) concepts in word recognition.

REFERENCES

[1] R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000 .

[2] S. Jaeger, S. Manke, J. Reichert and A. Waibel, "Online handwriting recognition: the NPen++ recognizer," *International Journal on Document Analysis and Recognition,* vol. 3, no. 3, p. 169–180, 2001.

[3] V. Carbune, P. Gonnet, T. Deselaers, H. . A. Rowley, A. Daryin, M. Calvo, . L.-L. Wang, D. Keysers, S. Feuz and P. Gervais, "Fast multi-language LSTM-based online handwriting recognition," *arXiv preprint arXiv:1902.10525,* 2019.

[4] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," in *Advances in neural information processing systems*, 2009.

[5] R. Safabakhsh and P. Adibi, "NASTAALIGH HANDWRITTEN WORD RECOGNITION USING A CONTINUOUS-DENSITY VARIABLE-DURATION HMM," *Arabian Journal for Science & Engineering (Springer Science & Business Media BV),* 2005.

[6] C. Olivier, H. Miled, K. Romeo and L. Yves, "Segmentation and coding of Arabic handwritten words," in *Proceedings of 13th International Conference on Pattern Recognition*, 1996.

[7] A. Lawgali, A. Bouridane, M. Angelova and Z. Ghassemlooy, "Automatic segmentation for Arabic characters in handwriting documents," in *18th IEEE International Conference on Image Processing, ICIP 2011*, 2011.

[8] A. Lawgali, M. Angelova and A. Bouridane, "A Framework for Arabic Handwritten Recognition Based on Segmentation," in *International Journal of Hybrid Information Technology*, 2014.

[9] I. Ahmad, L. Rothacker, G. A. Fink and S. A. Mahmoud, "Novel Sub-Character HMM Models for Arabic Text Recognition," in *12th International Conference on Document Analysis and Recognition*, 2013.

[10] I. Ahmad, G. A. Fink and S. Mahmoud, "Improvements in Sub-Character HMM Model Based Arabic Text Recognition," in *14th International Conference on Frontiers in Handwriting Recognition*, 2014.

[11] A. Graves, S. Fernández, F. Gomez and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML '06 Proceedings of the 23rd international conference on Machine learning*, 2006.

[12] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE,* pp. 257 - 286, 1989.

[13] A. Gimenez and A. Juan, "Embedded Bernoulli Mixture HMMs for Handwritten Word Recognition," in *10th International Conference on Document Analysis and Recognition*, 2009.

[14] P. Dreuw, P. Doetsch, C. Plahl and H. Ney, "Hierarchical hybrid MLP/HMM or rather MLP features for a discriminatively trained Gaussian HMM: A comparison for offline handwriting recognition," in *18th IEEE International Conference on Image Processing*, 2011.

[15] S. Abdel Azeem and H. Ahmed, "Effective Technique for the Recognition of Writer Independent Off-line Handwritten Arabic Words," in *International Conference on Frontiers in Handwriting Recognition*, 2012.

[16] M.-Y. Chen, A. Kundu and S. N. Srihari, "Variable duration hidden Markov model and morphological segmentation for handwritten word recognition.," in *IEEE Trans Image Process*, 1995.

[17] A. Graves, M. Liwicki , S. Fernández, R. Bertolami, H. Bunke and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[18] A. Graves, S. Fernández, M. Liwicki, H. Bunke and J. Schmidhuber, "Unconstrained On-line Handwriting Recognition with Recurrent Neural Networks," in *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, 2007.

[19] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," in *Advances in Neural Information Processing Systems 21*, 2009.

[20] B. Shi, X. Bai and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[21] A. Graves, A.-r. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE international conference on acoustics, speech and signal processing*, 2013.

[22] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks,* pp. 602-610, 2005.

[23] A. Graves and N. Jaitly, "Towards End-To-End Speech Recognition with Recurrent Neural Networks," in *International Conference on Machine*, 2014.

[24] A. Graves, Supervised sequence labelling with recurrent neural networks, 2012.

[25] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze and H. Amiri, "IFN/ENIT - DATABASE OF HANDWRITTEN ARABIC WORDS," in *7th Colloque International Francophone sur l'Ecrit et le Document*, 2002.

[26] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *arXiv:1502.03167*, 2015 .

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," 2014.

[28] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE* , 1998.

[29] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in neural information processing systems*, 2012.

[30] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[31] P. Y Simard, D. Steinkraus and o. C Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Document Analysis and Recognition, 2003. Seventh International Conference on*, 2003.

[32] M. D Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, 2014.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[34] A. Graves, S. Fernández and J. Schmidhuber, "Multi-Dimensional Recurrent Neural Networks," in *Artificial Neural Networks – ICANN 2007*, 2007.

[35] A. Graves, S. Fernández and J. Schmidhuber, "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition," in *Artificial Neural Networks: Formal Models and Their Applications*, 2005.

[36] S. Hochreiter, Y. Bengio, P. Frasconi and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," *A field guide to dynamical recurrent neural networks. IEEE Press,* 2001.

[37] M. Schuster and K. K Paliwal, "Bidirectional recurrent neural networks," in *IEEE Transactions on Signal Processing*, 1997.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* pp. 1735-1780, 1997.

[39] J. Sadri, M. R. Yeganehzad and J. Saghib, "A novel comprehensive database for offline Persian handwriting recognition," *Pattern Recognition,* pp. 378-393, 2016.

[40] E. El-Sherif and S. Abdelazeem, "A Two-Stage System for Arabic Handwritten Digit Recognition Tested on a New Large Database," in *International Conference on Artificial Intelligence and Pattern Recognition*, 2007.

[41] H. Khosravi and E. Kabir, "Introducing a very large dataset of handwritten Farsi digits and a study on their varieties," *Pattern Recognition Letters,* pp. 1133-1141, 2007.

[42] S. Mozaffari and H. Soltanizadeh, "ICDAR 2009 Handwritten Farsi/Arabic Character Recognition Competition," in *10th International Conference on Document Analysis and Recognition*, 2009.

[43] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu and X. Zheng, "TensorFlow: a system for large-scale machine learning," in *OSDI'16 Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, 2016.

[44] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning,* pp. 26-31, 2012.

[45] V. Margner and H. E. Abed, "Arabic Handwriting Recognition Competition," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007.

[46] A. Poznanski and L. Wolf, "CNN-N-Gram for HandwritingWord Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[47] A. Alaei , P. Nagabhushan and U. Pal, "Fine Classification of Unconstrained Handwritten Persian/Arabic Numerals by Removing Confusion amongst Similar Classes," in *10th International Conference on Document Analysis and Recognition*, 2009.

[48] A. Alaei , U. Pal and P. Nagabhushan, "Using Modified Contour Features and SVM Based Classifier for the Recognition of Persian/Arabic Handwritten Numerals," in *Seventh International Conference on Advances in Pattern Recognition*, 2009.

[49] A. Boukharoubaa and A. Bennia, "Novel feature extraction technique for the recognition of handwritten digits," *Applied Computing and Informatics,* pp. 19-26, 2017.

[50] F. Stahlberg and S. Vogel, "The QCRI Recognition System for Handwritten Arabic," in *Image Analysis and Processing — ICIAP 2015. ICIAP 2015. Lecture Notes in Computer Science*, 2015.

[51] I. Ahmad and G. A. Fink, "Handwritten Arabic text recognition using multi-stage sub-core-shape HMMs," *International Journal on Document Analysis and Recognition (IJDAR),* p. 1–21, 2019.