Temporal Order and Pen Velocity Recovery for Character Handwriting Based on Sequence-to-Sequence with Attention Mode

Temporal Order and Pen Velocity Recovery for Character Handwriting Based on Sequence-to-Sequence with Attention Mode

Besma Rabhi*[a], Abdelkarim Elbaati[b], Houcine Boubaker[a], Yahia Hamdi[a], Amir Hussain[c], Adel M. Alimi[a]

[a]*University of Sfax, National Engineering School of Sfax, REGIM-Lab.: REsearch Groups in Intelligent Machines, LR11ES48, 3038 Sfax, Tunisia*
[b]*University of Monastir, Higher Institute of Applied Sciences and Technology of Mahdia, REGIM-Lab.: REsearch Groups in Intelligent Machines, LR11ES48, 5121Mahdia, Tunisia*
[c] *School of Computing, Edinburgh Napier University, Edinburgh, United Kingdom*

ABSTRACT

Online signals are rich in dynamic features such as trajectory chronology, velocity, pressure and pen up/down movements. Their offline counterparts consist of a set of pixels. Thus, online handwriting recognition accuracy is generally better than offline. In this paper, we propose an original framework for recovering temporal order and pen velocity from offline multi-lingual handwriting. Our framework is based on an integrated sequence-to-sequence attention model. The proposed system involves extracting a hidden representation from an image using a Convolutional Neural Network (CNN) and a Bidirectional Gated Recurrent Unit (BGRU), and decoding the encoded vectors to generate dynamic information using a BGRU with temporal attention. We validate our framework using an online recognition system applied to a benchmark Latin, Arabic and Indian On/Off dual-handwriting character database. The performance of the proposed multi-lingual system is demonstrated through a low error rate of point coordinates and high accuracy system rate.

Keywords:Temporal order recovery; Pen velocity reconstruction; Deep learning; BGRU; Attention model

1. Introduction

Handwriting analysis has been an active area of research, such as handwriting recognition [11, 13], writer identification [8], and signature verification [7, 14]. When handwriting is captured using different acquisition techniques, it gives rise to two handwriting categories: online and offline. In case of online handwriting, it may require special digital devices and it can represent numeric data as a succession of points ordered in time. Consequently, this mono-dimensional signal is noted as having dynamic features: the temporal order, the pen velocity, the pressure, and the pen up / down. In contrast, offline handwriting requires a camera or a scanner to capture handwriting from the paper. Therefore, online devices are more expensive compared to the offline ones. However, it is obvious that online handwriting has become an efficient choice thanks to its dynamic features, thus authorizing more features to be available to the recognition systems. It is necessary to mention the importance of the pen velocity that develops the online information and to improve the recognition accuracy. In [29], the authors demonstrated the effectiveness of the velocity in the recognition task applied on Arabic handwriting characters. They obtained 98.8% for a re-sampled online signal against 95.8% for an online signal without velocity. In addition, considering that the offline category is a set of static images, the storage of handwriting images is larger than online data. Those images are presented by a set of pixels without dynamic information. Generally, the presence of dynamic features in online systems leads to the effectiveness of the performance compared to offline systems. Qiao et al. [28] proved the effectiveness of the online system compared to the offline one, using the recognition rates as an evaluation metric. They got 96% for online digits, against 90% for offline images. To exploit the advantages of offline and online processing, researchers have presented a lot of methods to recover the temporal order from static handwriting images. Reconstructing the drawing order has been implemented since the nineties [3, 5]. In general, this process is based on different steps: preprocessing, ambiguous zone selection, terminal point detection, and searching for the smoothest path [25]. According to Rousseau et al. [32], each step had an effect on the next step. Moreover, these last methods have suffered from assumption problems because the direction will be different according to the language of writing. In [25], the authors affirmed that the trajectory chronology proved to be more promising for reconstruction, but there was no way to recover some dynamic information like pen velocity. Actually, deep learning can handle these types of problems without using complicated algorithms or assumptions. Memory recurrent networks [17], with the potential to treat long term sequential tasks, realized great success. Among these investigations, image caption [37] achieved the level of translating images into text. Motivated by this, we assume that Sequence-to-Sequence (Seq2Seq) with attention models have a great potential to become the newly state-of-the-art for handwriting recovery problems. Our framework contains: a) a Convolutional Neural Network (CNN) to extract the lower level features, b) a Bidirectional Gated Recurrent Unit (BGRU) to encode the last extracted features to a single vector, and c) a BGRU to decode the encoded features into ordered coordinates based on attention model. To the best of our knowledge, we have been the first to implement a Seq2Seq-BGRU with attention model for temporal-order and pen-velocity recovery. The major contributions of this work are as follows:

- Investigating a novel Seq2Seq with attention model based on BGRU NN to predict the dynamic information from static handwriting images;
- Combining CNN and BGRU to extract features from images;

* Corresponding author. e-mail: besma.rebhi.2015@ieee.org.

- Recovering, for the first time, the pen velocity besides the temporal order;
- The ability of this end-to-end system to recover a multilingual character, so there are no assumptions about the pen order.

The rest of the paper is presented as follows. Section 2 sets an overview of related work. Section 3 describes the framework of the study. Section 4 discusses the implementation and the obtained results. Finally, section 5 provides the implications of the study and the conclusion.

2. Related work

Some work quoted in the literature has recovered the temporal trajectory order according to one category: a contour or skeleton approach. The contour technique [10, 34] has suffered from high computational time. For example, in [34], the authors were interested in loop analysis. They processed different models for loop types and performed a thorough loop contour analysis. Nevertheless, the effectiveness of the proposed investigation on loops was not clear enough as they did not show any practical results of handwriting recognition and since the valuation time was high. On the other hand, the use of the skeleton approach has given good results and a more rapid response compared to the contour method [9, 12, 21, 28, 32]. Based on the edge continuity relation [28], the authors suggested three main steps. First, they identified different relations at each node. For a node of degree four, they used the NN, otherwise they utilized some assumptions. Second, they selected double-traced lines using maximal weighted matching. Their last step was to find the smoothest possible path to go through all the curves of the handwriting graphic model. Based on the optimal Euler path, they selected the smoothest one. However, their work was applied on a single stroke for a mono language. In [32], the authors utilized the handwriting knowledge to propose the possible start / end points. Afterwards, different paths were produced and the best one was chosen. Their approach was applied on multi stroke letters. To demonstrate the performance of their approach, they presented a good recognition rate. Even so, they used the assumptions based on the Latin language only. In addition to the presence of contour and skeleton categories, surrounding areas of handwriting recovery could be divided into two groups: local and global search methods. The local tracing method goal was to search for the smoothest path at each ambiguous zone based on the tracing history and the actual configuration [3, 7]. The major limitation of this method was that the design of heuristic rules applied for different handwriting styles was difficult. This limitation could be overcome by using the global graph technique. It aimed to create a graph model of the input skeleton images and then use a search technique to find an optimal path through text [5, 12, 21, 28, 32]. The drawback of the global method was that the computational time was high and it depended on the complexity of the algorithms used as search techniques. There were also some cases which were hardly treated. For example, Phan et al. [26] used the greedy algorithm for searching about the optimal path in a global model. Their work was based on limited assumptions about start / end points, ambiguous zones and double traces segments, which gave rise to a difficult decision for obtaining the right trajectory. In [9], the authors considered that start point detection and skeleton separation represented a hard task. In addition, there was a higher complexity of searching the smoothest path at the junction zone. Based on the skeleton graph, they separated touching characters and crossing strokes. The optimal path was fixed by the greedy algorithm. Their model was still sensitive to the processed language, and they had the common problem of being slow and complex [19, 28, 32]. However, it was not clear whether the use of the local method could achieve a more effective performance compared to the global method. Both suffer from some problems. Consequently, some existing work has combined these two tracing methods together [12], where the number of possibilities was optimized by adding some local features such as the curvature and inclination angle. It cannot be denied that previous work has got perfect performances, particularly in the Latin language. However, most of them have been weakened by some languages, like the Arabic handwriting corpus, hence having many versions for each language [12]. Moreover, the problem of handwriting recovery has been based on finding out the correct terminal points (start/end points), junction points and the main direction in the detected ambiguous zone. Furthermore, Rousseau et al. [32] demonstrated that each step of the recovery procedure could affect the results of the recognition rate. Thus, in our opinion, these challenging parts were complicated. Some early work [23, 29] addressed the use of an end-to-end system for handwriting recovery. In [29], the authors used VGG-LSTM to extract features from images and utilized BLSTM as a decoder model. Their system was the most closely related work. In our work, we refer to [29] where the focus was different. We use CNN-BGRU to extract features from images, instead of VGG-LSTM. In the task of recovering the temporal order from offline handwriting, it is more challenging to produce human-like velocity. Indeed, the authors in [29] recovered an online signal with equidistant points. They utilized a re-sampling step to add velocity to the obtained signal. However, in this paper, our framework produces an online signal characterized by a trajectory chronology with velocity.

3. End-to-end recovery framework

In this section, we clarify the main architecture of the proposed framework. We employ a Seq2Seq with attention model [2] to transform a sequence of offline handwriting characters into a sequence of their homologous online signal. The obtained signal contains dynamic features like the temporal order and the pen velocity. The main objective of this work is shown in the following equation:

$$P[pc|fp] \tag{1}$$

where *pc* is the generated sequence of point coordinates characterized by dynamic features. Those points correspond to image I. Moreover, *fp* is a sequence of pixels which represent a static feature of I. As a result, the length and type of the input (image) and output (signal) are actually different. Our model consists of three parts: a CNN, an encoder BGRU NN and a decoder with attention model. These parts are considered as an end-to-end system. Therefore, training is supervised, taking into account image I and its counterpart online signal points *n*, i.e. $\{I\}= \{(x_1,y_1),\ldots,(x_n,y_n)\} \in R^{2*n}$. Fig. 1 represents an overview of the proposed framework. The ConvExtractor function C() is a CNN that transforms each image I into features F. The input images are embedded and converted to a vector. These features are extracted according to the sliding receptive field across the image from left to right (from top to bottom).

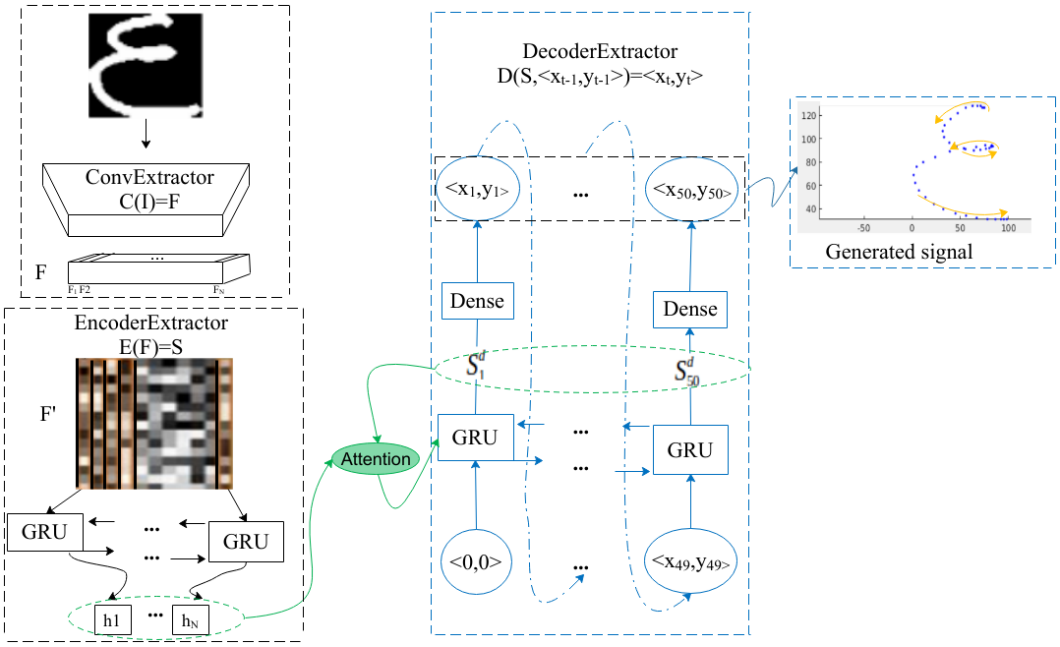


Fig. 1. Architecture of Seq2Seq BGRU model. For sake of simplicity, one BGRU layer is shown.

The ConvExtractor receives offline handwriting and produces a sequence of features associated with the most significant part of pixels forming the letter of the image. The EncoderExtractor function E() is a multilayer BGRU model that accepts the sequence of features F and extracts the last state S. The attention layer is placed between the encoder and the decoder extractors to highlight some effective encoder output features. Finally, the DecoderExtractor D() function is a multilayer BGRU model. It receives the local context state generated by the attention layer and the previous predicted coordinates. In section 4, we will show the effectiveness of different basic encoder-decoder models and prove that a BGRU NN is the best candidates. To summarize, the following equations present the recent processes:

$$F=C(I) \tag{2}$$

$$S=E(F) \tag{3}$$

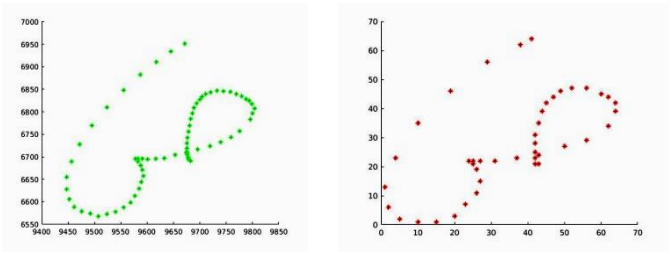
$$(x_t,y_t)=D(S, <x_{t-1},y_{t-1}>) \tag{4}$$

where $<x_{t-1},y_{t-1}>$ are the previously predicted points.

3.1. Preprocessing and pen velocity process

The training step is based on two inputs: a) the handwriting character image, and b) its counterpart online signal. Generally, the first step in handwriting recovery is preprocessing. This process can include image normalization, taking into account that all images have the same size (64 x 64). Those images are passed to the ConvExtractor C(). Their counterpart online signals are used as an input to the DecoderExtractor D(). This type of signals is necessary to train the supervised decoder BGRU network. The number of online signal points is fixed to 50 points. A sampling step is necessary to obtain a signal with the desired point number (see the algorithm of normalization step). In fact, online handwriting is a series of non equidistant points saved during the writing process. A study made on the neuronal and muscular effect shows that the pen velocity decreases at the terminal points of strokes and at the junction zone of the curve. This information is used in online systems to calculate the velocity. Thus, those points are presented as a two-dimensional matrix and characterized by the pen velocity. To the best of our knowledge, researchers have not reconstructed the pen velocity when solving the handwriting recovery problem. However, in [12, 29], the authors used the re-sampling step to add the velocity to the recovered signal, without generating a significant signal with temporal order and pen velocity at the same time. By way of contrast, our framework does that.

Fig. 2 (a) shows the ground truth signal. The speed of the pen decreases at the beginning, at the end of strokes and at the insignificant angular zones of the curve. This type of signals is used as an input to the algorithm of the normalization step. Fig. 2 (b) represents the output signal after normalization according to the number of points (50 points), while maintaining the same speed as the original. The 50 points are normalized to (64*64).



(a) (b)

Fig. 2. Example of online signal of Arabic letter /sa:d/. Green color indicates the original online signal (a), red color represents the normalized signal (b).

The GRU and LSTM networks can be adapted to different sizes (with an end-of-stroke token). However, the proposed framework treats 50 points as [23, 28], because each point will be more informative and the long dependencies will be reduced. The normalization step keeps the velocity information consistent within a character, and the distance between two points will depend on the natural speed of the pen and on the total length of the character. As a consequence, the strength of the following algorithm is to solve this issue and normalize the online signal with a fixed point number.

Algorithm: Normalization step with fixed point number

Input: - Matrix of original trajectory M_I(j) of size N_I
- Number of points N_O of expected normalizing

Output: -Matrix of the normalized trajectory M_O(i) of size N_O

1: Calculate curvilinear length Cl of original trajectory

2: Calculate trajectory average speeds obtained in original and expected resampling, respectively:
Vm_I = Cl / (Δt. N_I); Vm_O = Cl / (Δt. N_O);

3: Calculate curvilinear abscissa x_curved_I(j) of each N_I (distance from start point to asked point)

4: Initialize current point of resampled trajectory: M_O (1,:) = M_I (1,:)

5: Initialize curvilinear abscissa of current point of resampled trajectory: x_curved_O(1) = 0

6: Initialize instantaneous speed / average speed proportionality factor at current point: p_f = 1; j=2
7: **For**: i = 2 **to** (N_O)
7.1: Calculate curvilinear abscissa of new point
x_curved_O(i) = x_curved_O(i-1) + (Vm_O. P_f. Δt)
7.2: Position new current point M_O (i) between two successive points M_I (j) and M_I (j + 1) of original sampling:
If (x_curved_O(i) >= x_curved_I(j)) AND (x_curved_O(i) <= x_curved_I(j+1)) OR (j+1=N_I)
P_f= |x_curved_I(j+1) - x_curved_I(j)| / (Vm_I. Δt);
Else j = j + 1
End if
7.3: Calculate local shift between original and expected sampling [M_O(i+1), M_I(j)]:
D_Mo_Mi = x_curved_O(i) - x_curved_I(j)
7.4: Calculate the current elementary shifting in the original sampling [M_I(j+1), M_I(j)]:
D_Mi_Mi = x_curved_I(j+1) - x_curved_I(j)
7.5: Caculate coordinates of new current point:
x_O_current_M = ((x_i_M(j,1)*(D_Mi_Mi-D_Mo_Mi)) + (x_i_M(j+1,1) * D_Mo_Mi)) / D_Mi_Mi
y_O_current_M = ((y_i_M(j,2)*(D_Mi_Mi-D_Mo_Mi)) + (y_i_M(j+1,2) * D_Mo_Mi))/D_Mi_Mi
7.6: Add new point M_O (i + 1) to matrix of points of re-sampled trajectory
End for
End

3.2. Convolutional extractor

Deep CNNs [24] have been used in different tasks [19, 23, 37]. Given the CNN success in providing a solid representation of features, we use CNNs extensively to extract a sequence of image features. The main goal of the convolutional extractor is to convert the character image into a visual feature. A deep CNN model is used without its last fully connected layer. The normalized images are fed into the network. Then, the convolutional layers produce the feature maps. The feature is extracted from left to right, and column by column, from the feature maps. One feature vector corresponds to a rectangular region. Those features describe the image region. The details of the CNN configuration are described in section 4. The input image (64*64) is transformed to CNN features of size (Batchsize, N, D). Specifically, the batch size is fixed to 32, and N and Dare the length and depth of the CNN entity, respectively. As illustrated in Fig. 1, the CNN output is noticed by F = (F_1, F_2, ..., F_N) and $F_i \epsilon R^D$ (D = 512).

3.3. Encoder extractor

The encoder-decoder model is previously employed for machine translation [2, 35], but recently it has been applied to other tasks [19, 23, 31, 37, 20]. The encoder is the portion of the network that processes the input to produce a single hidden representation of all the input information. Among the varied types of neural nets, Recurrent NNs (RNNs) can forecast the most accurate results [15, 30]. In fact, the famous problem of simple RNNs is the vanishing gradient problem, which the GRU [6] and LSTM [16] can alleviate. According to previous studies [18], there were no conclusions that could be drawn about the best: LSTM or GRU. Both RNNs have achieved similar results in many tasks. Even so, the GRU can be better in terms of time thanks to the parameter size. In addition, according to [17], the GRU could be a better choice for modeling the temporal information compared to the LSTM. That is why the GRU is chosen instead of LSTM. The hidden state is computed by the following equations:

$$g_t=\sigma(W_{fg}F_t+U_{hg}h_{t-1}) \quad (5)$$

$$r_t=\sigma(W_{fr}F_t+U_{hr}h_{t-1}) \quad (6)$$

$$c_t=tanh(W_{fh}F_t+U_{rh}(r_t*h_{t-1})) \quad (7)$$

$$h_t=(1-g_t)*h_{t-1}+g_t*c_t \quad (8)$$

where g_t , r_t and c_t are respectively the updated gate, the reset gate and the candidate activation value, σ is the sigmoid function, F_t is the input which is the extracted CNN features, and W_{fg}, U_{hg}, W_{fr}, U_{hr}, W_{fh} *and* U_{rh} represent different weights. To obtain more information and a better representation, the BGRU with multiple layers is chosen as the most efficient encoder type adapted to our problem. Precisely, after many experiments of various models with different configurations, the encoder is fixed as a BGRU NN with three layers, and each layer has 512 units. The encoder BGRU model is built after obtaining the CNN features extracted from the image character. This feature vector is mapped to a fixed-length vector through the encoder. As depicted in Fig. 1, the ConvExtractor generates an intermediate feature F which is reshaped to an F' map (F'_1, F'_2, …, F'_N) with two dimensional features. Those features are used as an input to the first layer of the BGRU encoder. The remaining layers use as an input their previous hidden state. For each time step, the output of the encoder is $h_t \epsilon$ H calculated by equation (8).

3.4. Decoder extractor

The decoder is a BGRU NN with three layers as the encoder architecture. Each layer has 512 units. The role of the DecoderExtractor is to generate the predicted coordinate sequences, as shown in equation (9):

$$P(P_1,\dots,P_i|F_1,\dots,F_N)=\prod_{j=1}^{i}P(P_i|S,P_1,\dots,P_{j-1}) \quad (9)$$

with $P_1, \dots, P_i = [<x_1,y_1>,\dots,<x_i,y_i>]$, and where i is the number of the desired points and F is the set of features. Besides, $F_N \epsilon R^D$ where N and D are respectively the length and depth of the CNN. According to the entire feature sequence (F_1,..., F_N), the decoder estimates a set of points ($P_1,\dots,P_i$). Moreover, S is the last encoder state that summarizes the whole input feature. The hidden state of the decoder is initialized with S at the same time step. In the first layer, the decoder takes as an input the last state of encoder S and the first coordinate <0,0>. Then, the remaining layers receive as an input the hidden state of the previous decoder layer S_{i-1}^d and the previous coordinate $<x_{i-1},y_{i-1}>$, as shown in equation (10):

$$S_i^d = g(S_{i-1}^d , [P_{i-1}, S]) \quad (10)$$

where S_i^d is the current state of the decoder, S_{i-1}^d is its previous state, and g is a GRU function. The generated coordinates are calculated by equation (11) as follows:

$$P(P_i|S,P_1,\dots,P_{i-1}) = dense(US_i^d + b) \quad (11)$$

Here, we apply a linear activation function (*dense*). Actually, U and b are the weights and bias, respectively.

As mentioned above, we introduce the recovery process of a classical Seq2Seq [35], which can resolve the problem of handwriting recovery.

3.5. Attention mechanism

The classic Seq2Seq without attention model has defects that all hidden representation of the encoder are compressed into a fixed length S context vector. Consequently, the prediction accuracy will gradually decrease when the input length increases [2]. Thus, this paper proposes an attention mechanism for the handwriting recovery process. An attention layer is placed between the encoder layer and the decoder. The input of the BGRU encoder is $F' = (F'_1, F'_2, \dots, F'_N)$. At each time step N, the encoder reads F'_N and updates its hidden state h_t Then, the attention context vectors are produced as a weighted sum of h_t,

which is used to detect the best hidden representation of the encoder. The following equation describes the attention process:

$$e_{i,t}=Align(S_{i-1}^{d}, h_t)= S_{i-1}^{d} \odot h'_t \quad (12)$$

$$\alpha_{i,t}=\frac{\exp(e_{i,t})}{\sum_{l=1}^{N}\exp(e_{i,l})} \quad (13)$$

$$c_a=\sum_{t=1}^{N}\alpha_{i,t}h_t \quad (14)$$

where formula (12) is the align computation between the encoder hidden state h_t and the decoder hidden state S_i^d. Formula (13) represents the attention weights that indicate the importance of the input value at time step *t* to generate the output at time step *i*. The softmax function is used to normalize the vector e_i (of length *N*) as the attention mask on the input sequence. Formula (14) shows the final attention state c_a. At each time step, the decoder predicts a probability distribution of point coordinates. The ground truth point coordinates are used to train the network how to generate an online signal compatible to the original script (see Fig. 2 (b)). The values of predicted point coordinates are updated according to loss *L1* as in equation (15):

$$L_1=\frac{1}{N}\sum_{t=1}^{N}|P(t)-P'(t)| \quad (15)$$

Let the ground truth vector be $P'= [<x'_1,y'_1>,...,<x'_i,y'_i>]$ where i is the number of points. We calculate loss *L1* using the predicted vector *P*. The training continued until loss *L1* converges and the model is saved. The test step uses the offline handwriting image as an input; and based on the obtained model, the framework can generate a set of point coordinates representing human-like writing.

4. Experiments

In this section we study the effect of different settings of basic encoder and decoder candidates. Then, we explore the performance of the proposed model and we compare our results with the existing methods on different datasets.

4.1. Datasets

We test the proposed framework on Arabic, Latin and Indian corpora. Specifically, for Arabic, we use the dual on / off Arabic LMCA dataset [22]. It contains 28 letters, which are joined or isolated (we choose the isolated form). For Latin data, we adopt the dual on / off Latin IRONOFF dataset [36]. We use the isolated letters (upper and lower cases) and digits, sorted into 26 and 10 classes, respectively. For Indian, we use the Telugu[1] dataset. It contains 116 Telugu characters. All these datasets are used without considering the pen up / down. We created additional patterns using a data augmentation strategy based on distorted samples (changing angle inclination, smoothing, and baselines). The obtained signals are converted to offline handwriting. First, we concatenate the pen points to obtain the skeleton of the image. Then, we use a filter to grow the skeleton so that if the current point is foreground, all its neighbors are set to the foreground. Table 1 represents the number of samples used for training, testing and validation.

Table 1. Dataset details

Scripts	Training	Testing	Validation
LMCA	70,000	20,000	10,000
IRONOFF Lower-case	56,000	17,800	10,200
IRONOFF upper-case	56,000	17,800	10,200
IRONOFF Digits	70,000	20,000	10,000
Telugu	58,000	17,400	11,600

[1] http://lipitk.sourceforge.net/datasets/teluguchardata.htm

4.2. Metrics and implementation

Thanks to the available on / off data for each corpus, the evaluation step becomes easy. Thus, both signals (truth and generated) are not only compared by the Root Mean Square Error (RMSE) and the Euclidean Distance (ED), but also recognized by an online recognition system. We state the effectiveness of our proposed framework on three evaluation metrics: the RMSE, the ED and the online recognition system based on the LSTM NN. The RMSE and the ED are chosen as evaluation criteria for distance parsing.

The RMSE is used to measure the rate of transformation from one set of points to another. Its definition has been used differently in related work such as [7, 14]. In our case, it represents the difference between the online signal and the recovered one according to the following formula:

$$\boldsymbol{RMSE}=\frac{\mathbf{1}}{\mathbf{2}\boldsymbol{L}}\left(\sum_{i=1}^{n}\sum_{t=1}^{li}(\boldsymbol{x_t}-\boldsymbol{x'_t})^{\mathbf{2}}+\sum_{i=1}^{n}\sum_{t=1}^{li}(\boldsymbol{y_t}-\boldsymbol{y'_t})^{\mathbf{2}}\right) \quad (16)$$

where *n* is the number of samples, l_i is the number of points in each sample, x_t and y_t are the coordinates of the online signal, x'_t , y'_t represent the recovered coordinates, and *L* denotes the total length of characters. The better system is the one that reaches the lower RMSE value. Different results are represented in Table 4.

The chosen ED is used in [7], as the following formula:

$$ED=\sqrt{(x_p-x'_q)^2+(y_p-y'_q)^2} \quad (17)$$

where *p* and *q* vary between *1* and *L*. The ED metric minimizes the cumulative distance calculated between the ground truth and the predicted elements via a warping path. Thus, the best method is the one that achieves the lowest values. The results are given in Table 4.

We agree that the evaluator metrics, the RMSE and even the ED, are brutal when evaluating a predicted signal with a small deviation, even if the temporal order is compatible with its ground truth. Nevertheless, the accuracy result could be unexpected. After this interpretation, we use an online LSTM recognition system [29], based on the beta elliptic and grapheme segmentation method [4] which requires the existence of the temporal order and the velocity to produce a reasonable result. We extract 10 characteristics from the original online signal to train the network. Then, we test the network with our reconstructed signal.

Our framework is trained over one Nvidia GT 650M with GPU in a Tensorflow platform. The training batch has 32 image-signal pairs. To update parameters, we choose Adam Stochastic with 10^{-3} of the learning rate. We save the checkpoint model every 700 iterations. Training time lasts for 16 hours and testing takes around seven minutes for each 20 samples.

4.3. Ablation study

Recently, there have been many networks used to extract features from images: ResNet-50 [39], VGG-16 [33] and CNN, followed by LSTM NN or GRU NN, to obtain a higher feature representation. Thus, we evaluate the encoder combinations on the IRONOFF digit dataset, and the results of the training loss are presented in Table 2. This table shows the following studies: (1) We use 5 networks (i.e. ResNet , VGG , CNN1, CNN2 and CNN3) followed by LSTM as an encoder. The CNN configurations are listed in Table 3. We fine-tune the pre-trained deep models (ResNet and VGG). The results demonstrate that with the same followed network (i.e. LSTM), CNN2 performs better than other networks in terms of training accuracy. We use

eight convolution layers with a kernel size of (3 * 3). The primary role of the Max pooling is to extract the main significant characteristics from the output of the previous convolution layers.

Table 2. The training loss of different encoder combinations on IRONOFF digits dataset

Models	Training accuracy
LSTM-V16	90.0
LSTM-R50	90.2
LSTM-C1	90.3
LSTM-C2	93.2
LSTM-C3	92.0
GRU-C2	**93.5**

Table 3. CNN configurations

		Parameter values for each layer
CNN1	**filters**	64 – 128 – 256 – 256 – 512 - 512
	Pool	(2,2) – (2,2) - No -(1,2) –(2,1)-No
CNN2	**filters**	64 – 128 – 256 – 256 – 256 – 256-256-512
	Pool	(2,2) – (2,1) - No -(2,1) –(2,1)-No-(2,1)-No
CNN3	**filters**	50 –100–150–200–250 – 300-350-400-512
	Pool	No – (2,2) -No -(2,2) –No-(1,2)-No-(2,1)-No

Each layer is preceded by an activation function (rectified linear unit) [1]. Batch normalization is employed after the third and last Conv. layers. (2) With CNN2, the results show that the GRU NN is better than LSTM. GRU-CNN2 is the best combination that achieves the lowest training loss. We investigate whether the proposed framework can generate the closest signal to the original. With the same GRU-CNN2 based encoder, we use five decoder-based combinations (i.e. Seq2Seq-LSTM, Seq2Seq-GRU, Seq2Seq-BGRU, and Att-BGRU).

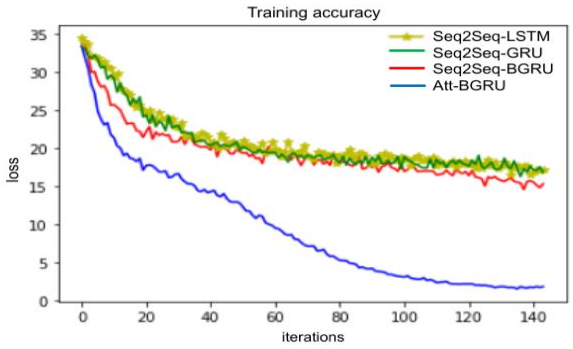


Fig. 3. The performance of different decoder combinations on IRONOFF digits dataset

Fig. 3 depicts the performance of different decoder combinations in terms of training accuracy. The results show that BGRU performs better than LSTM and GRU for the Seq2Seq model. In addition, the attention model achieves an accurate result compared to simple Seq2Seq. In other words, we choose GRU-CNN2 as the encoder and Att-BGRU as the decoder for handwriting recovery.

Table 4. A comparison of the average error (RMSE and ED) and the recognition rate of our framework Att-BGRU-V with other baseline frameworks on five datasets

Models	IRONOFF lower-case			IRONOFF upper-case			IRONOFF digits			LMCA			Telugu		
	RMSE	**ED**	**%**	**RMSE**	**ED**	**%**	**RMSE**	**ED**	**%**	**RMSE**	**ED**	**%**	**RMSE**	**ED**	**%**
Att-BGRU	3.1	31.9	93.0	2.1	23.1	94.3	1.9	19.8	94.9	2.0	20.0	98.9	2.5	23.1	93.2
S2S-BLSTM-V	3.5	32.1	90.0	2.8	23.8	90.5	2.2	20.3	89.5	2.3	21.2	92.0	3.1	24.1	89.5
S2S-VBLSTM-V	3.4	32.0	92.9	2.2	23.7	94.2	2.1	20.0	94.6	2.1	21.0	98.9	3.0	23.6	93.0
S2S-CBGRU-V	3.0	31.9	93.1	2.1	23.0	94.4	2.0	19.8	95.6	2.2	20.0	98.9	2.7	23.1	93.1
Att-BGRU-V(Ours)	**2.9**	**30.0**	**93.2**	**2.0**	**22.8**	**94.6**	**1.8**	**19.7**	**95.7**	**1.8**	**21.9**	**98.9**	**2.3**	**23.0**	**93.4**

Table 5. A comparison of the average error (RMSE and ED) and the recognition rate of our framework Att-BGRU-V with the state-of-the-art methods on five datasets

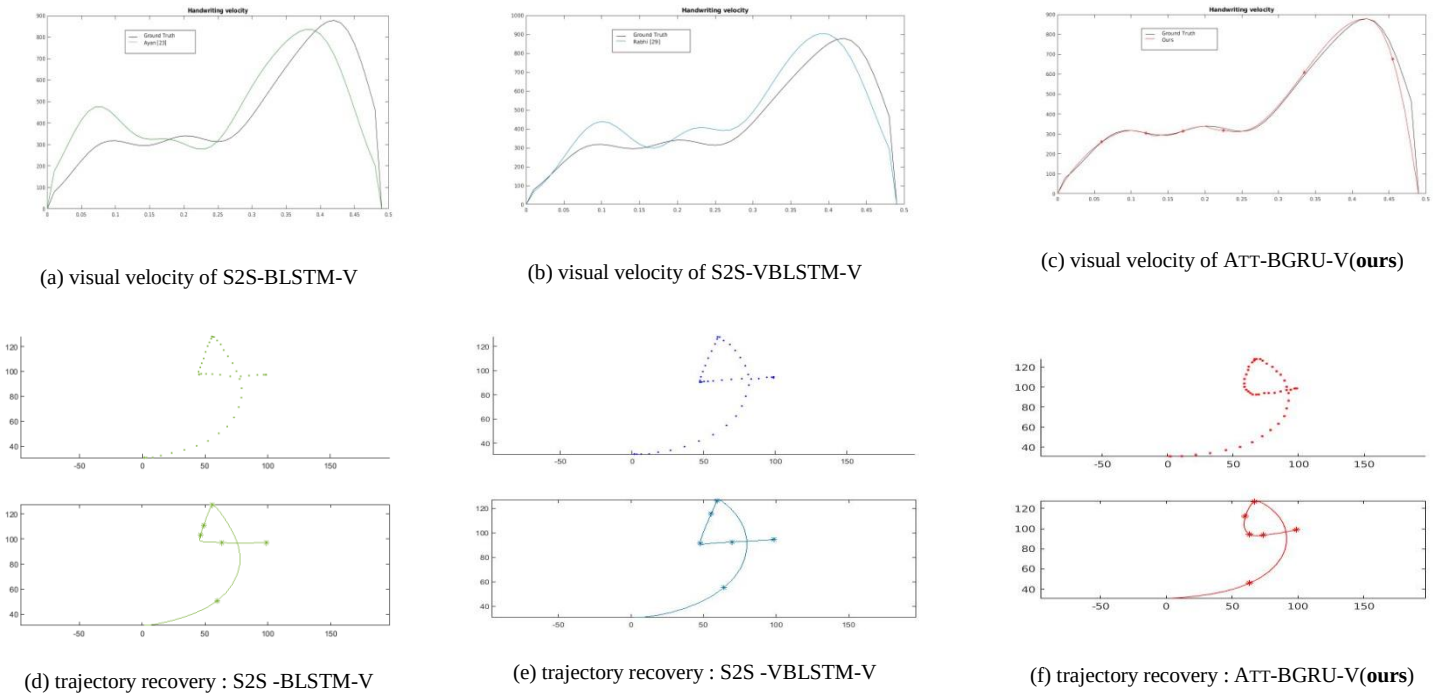
Models	IRONOFF lower-case			IRONOFF upper-case			IRONOFF digits			LMCA			Telugu		
	RMSE	**ED**	**%**	**RMSE**	**ED**	**%**	**RMSE**	**ED**	**%**	**RMSE**	**ED**	**%**	**RMSE**	**ED**	**%**
Elbaati [12]	20.1			19.0			18.0			32.9			20.1		
S2S-BLSTM[23]	3.6	32.2	88.9	2.9	23.9	90.4	2.3	20.3	89.3	2.5	21.3	91.9	3.2	24.2	89.2
S2S-VBLSTM[29]	3.5	32.1	92.8	2.3	23.8	94.1	2.1	20.1	94.5	2.2	21.2	98.8	3.1	23.7	92.9
Att-BGRU-V(Ours)	**2.9**	**30.0**	**93.2**	**2.0**	**22.8**	**94.6**	**1.8**	**19.7**	**95.7**	**1.8**	**21.9**	**98.9**	**2.3**	**23.0**	**93.4**

4.4. Architecture exploration

We aim to demonstrate whether the proposed framework and velocity reconstruction can boost the performance of the handwriting recovery process. Specifically, we compare five different handwriting recovery models: (1) the basic model (Att-BGRU) presented in section 4.3, which utilizes equidistant points as online training data, so it is described as a framework without velocity; (2) sequence-to-sequence based BLSTM with velocity (S2S-BLSTM-V), which is inspired from [23] and which integrates the velocity feature. Moreover, training online data is passed by the algorithm of normalization (introduced in section 3.1) to obtain an online script with velocity and a fixed point number; (3) sequence-to-sequence with a VGG-16 and BLSTM based encoder (S2S-VBLSTM-V), which is inspired from [29] and takes the similar training data as S2S-BLSTM-V; (4) sequence-to-sequence with a CNN2 and BGRU based encoder (S2S-CBGRU-V), which takes similar training data as the latter model; and (5) our Att-BGRU-V framework which integrates the velocity concept. as detailed in section 3.1 .

All the experimental results are shown in Table 4. From Table 4, we can see the following: (1) BGRU with the attention model (Att-BGRU) performs better than both S2S-BLSTM-V and S2S-VBLSTM-V. This indicates that the attention mechanism can improve the performance of handwriting recovery. However, in some cases, it has low margin effectiveness compared to S2S-CBGRU-V. This demonstrates the effectiveness of BGRU trained by signals with velocity, so, the pen velocity reconstruction proves its effectiveness. (2) S2S-CBGRU-V performs slightly better than both S2S-BLSTM-V and S2S-VBLSTM-V, thanks to the BGRU NN which outperforms BLSTM for handwriting recovery. (3) The results show that our Att-BGRU-V achieves the best performance over all the evaluation metrics (i.e., 2.0 RMSE, 22.8 ED and 94.6% recognition rate) on the IRONOFF upper-case. (4) We show the effectiveness of the pen velocity process and we assess the model on another scheme without velocity (Att-BGRU), which is obtained after training the framework with offline handwriting and its corresponding online one (a set of equidistant points). By comparing between the latter and our proposed framework (Att-BGRU-V) with velocity, we find that using the pen velocity is very efficient for handwriting recognition parsing.

To sum up, the experimental results demonstrate that the proposed framework with the jointly attention model and BGRU with velocity enhances the effectiveness of the handwriting recovery process.



(a) visual velocity of S2S-BLSTM-V

(b) visual velocity of S2S-VBLSTM-V

(c) visual velocity of ATT-BGRU-V(**ours**)

(d) trajectory recovery : S2S -BLSTM-V

(e) trajectory recovery : S2S -VBLSTM-V

(f) trajectory recovery : ATT-BGRU-V(**ours**)

Fig. 4. A comparison of our reconstructed velocity with two existing models

4.5. Comparison to existing methods

To evaluate the effectiveness of the proposed framework, we compare our work with three other located systems, which have been re-implemented and tested under the same environment conditions. Our proposal consists in using the CNN2-BGRU based encoder and the Att-BGRU based decoder. This model is intended to recover an online signal with velocity. Precisely, we compare four frameworks :(1) Elbaati's approach [12] was based on the graph model to represent an image as a set of segments. The genetic algorithm was used to find the smoothest path across those segments (2) S2S-BLSTM proposed in [23], was the classical Seq2Seq without the attention model with CNN-BLSTM as an encoder and BLSTM as a decoder. (3) S2S-VBLSTM proposed in [29], was similar to S2S-BLSTM but with a VGG-16 and BLSTM based encoder, and the authors applied the sampling step on the recovered signal to add velocity. (4) Our baseline model (Att-BGRU-V) is introduced in section 4.4. All the experimental results are provided in Table 5.

From Table 5 , we can admit that:

(1) The approach of Elbaati et al. [12] had a low margin effectiveness compared to all deep models and over all the evaluation metrics. This indicates that with the rise of deep learning, handwriting recovery can be handled more efficiently.

(2) The framework of Ayan et al. [23] was the first deep model for handwriting recovery. Their higher accuracy rate (91.9%) can be achieved when testing the LMCA dataset, but it will remain below both the S2S-VBLSTM [29] rate (98.8%) and our rate (98.9%). This indicates that pen acceleration obtained after a sampling step [29] proves its performance. In addition, the attention model (ours) with velocity has the advantage of improving the recognition accuracy.

(3) Rabhi's framework [29] was the best state of the art framework. The authors re-sampled the recovered signal (set of equidistant points) and obtained an online signal with pen acceleration. However, our framework generates a significant signal with velocity without a post-processing step based on a sampling step. As a result, they achieved a lower accuracy compared to ours, thanks to the attention mechanism which can handle the performance. In addition, we can interpret that the BGRU NN proves its efficiency in terms of accuracy and time (0.7s/step) compared to BLSTM (1s/step).

(4) Our proposed Att-BGRU-V performs better than the state-of-the-art models in terms of all evaluation metrics. It outperforms Elbaati's approach [12], Ayan's system [23] and Rabhi's framework [29] in 16, 0.3 and 0.1 absolute error points, respectively, when testing the data of digits. For the ED evaluation, it can be affirmed that our proposed framework achieves the best performance. It surpasses Elbaati's approach [12], Ayan's system [23] and Rabhi's framework [29] in 30.9, 0.2 and 0.1 absolute error points, respectively, when testing the Arabic data. In addition, the recognition rate, whatever the database, is higher than other methods.

To sum up, the effectiveness of the velocity is captured when we apply the attention mechanism with BGRU NN.

Visual analysis of the reconstructed velocity. We analyze the velocity prediction of the proposed framework using a visual graphic on the Arabic letter <<waw>> from the LMCA dataset. Fig. 4 (a)-(c) shows the velocity curves of both ground truth and predicted models (S2S-BLSTM-V, S2S-VBLSTM-V and our model ATT-BGRU-V). As shown in these figures, the difference of deviations between the predicted velocity of different models and the ground truth velocity are not too large. Our model has the closest deviation to the ground truth. To further analyze the reconstructed velocity, Fig. 4 (d)-(f) shows the trajectory reconstruction corresponding to the latter curves. These trajectories are divided into strokes based on the inflection points which are located according to the variation of pen acceleration [4]. As indicated in these figures, the difference between the location of points is not interesting. However, our model reconstructs a trajectory with a flexible curvature as human writing, thanks to the attention layer which can focus on the detailed oval curves.

Analysis of reconstructed characters. Fig. 5 presents two successful recovered samples of an Indian character and an IRONOFF digit. These scripts are reconstructed successfully using the models with and without an attention layer.

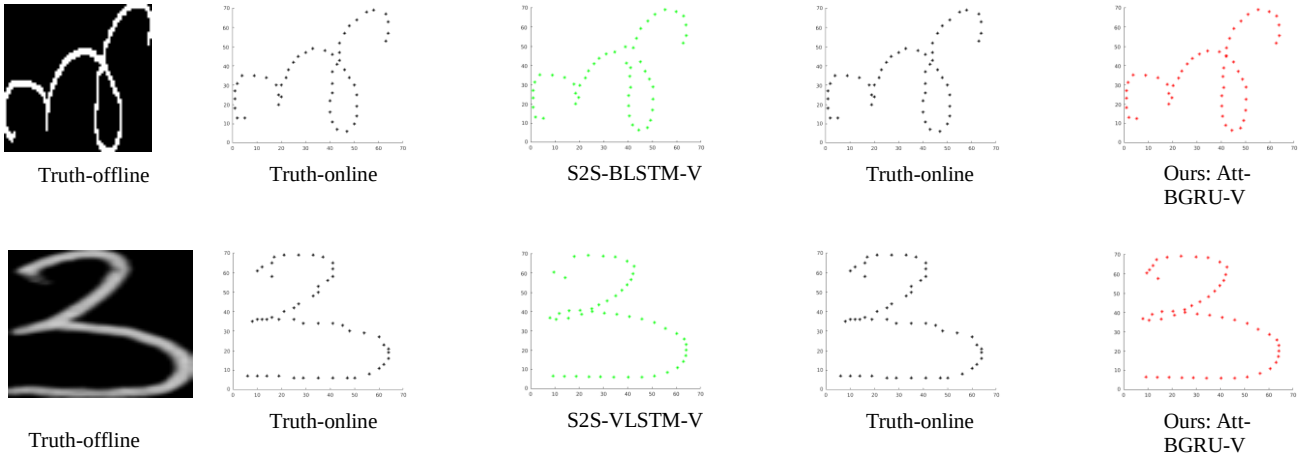


Fig. 5. Successfully reconstructed characters

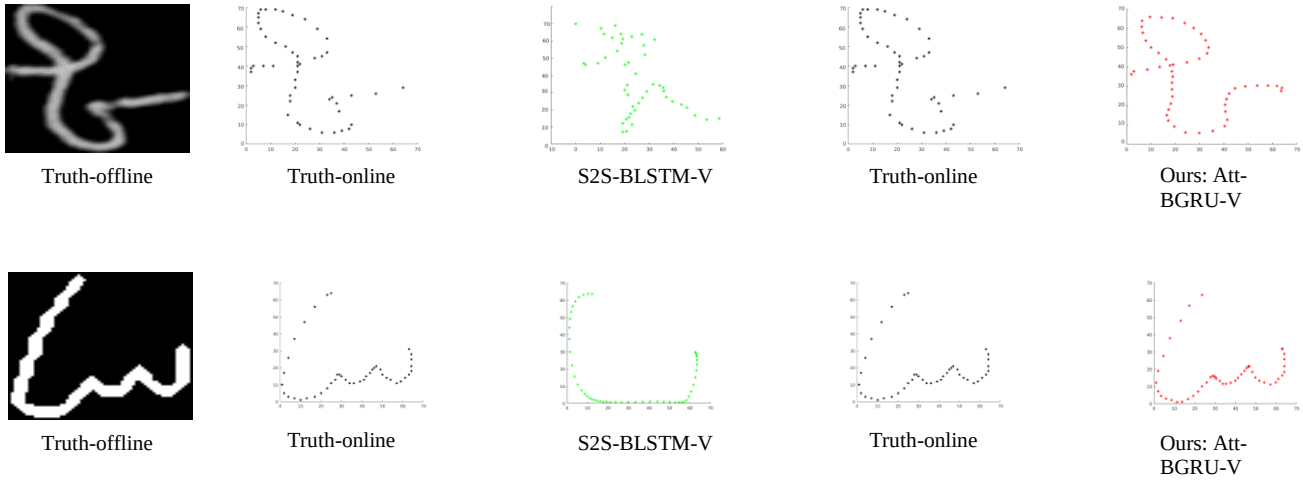


Fig. 6. Successfully reconstructed characters with attention model

Despite that, the models without attention (S2S-BLSTM-V, S2S-VBLSTM-V) could recover the online trajectories which are not matched to the offline image. because the encoder-decoder model extracts the final encoder state from different character samples. due to the absence of the attention layer, the decoder generates an online trajectory which can be the same as some existing samples in the training dataset (see Fig. 6). Consequently, the attention layer can avoid overfitting and adapt to truth samples.

Fig. 6 shows two reconstructed samples of Latin and Arabic characters. These samples are successfully reconstructed with our attention framework and fail with models without attention. In some cases, the latter model generates erroneous scripts which are different to the counterpart offline handwriting. These scripts could refer to some existing samples of the target dataset. For example, when recovering the character << sin >>, we obtain an erroneous signal which refers to the existing character << ba >>. In addition, the Latin letter << b >> is recovered as character <<z>> which exist in the IRONOFF dataset. Here, the decoder refers to the last encoder feature which can be similar to existing encoder feature of other samples.

Fig. 7 shows our recovered signal with a small deviation compared to its counterpart ground truth signal. Simultaneously, the temporal order is suitable for the online one. The proposed framework uses as an input an offline image and generates an online signal with temporal order and pen velocity. Fig. 8 represents the pen velocity of our recovered signal and the truth signal.

Acceleration decreases at the zones of the red circles similarly to the ground truth signal. This makes us admit that the recovered signal respects the velocity like human writing. As indicated in Fig. 9, the recovered signal passes the loop in the truth direction. Both start and terminal points are basically compatible to the original one. Thus, the temporal order of our recovered signal respects those of the truth signal. Fig. 10 depicts some cases of the suggested framework where the zone marked by a pink arrow is wasted because of the up / down pen. This process has not been treated yet. In fact, the proposed system deals with mono-stroke isolated characters. Our contribution focuses on the reconstruction of the pen velocity feature, which has not been treated yet, neither with old systems nor with new ones.

4.6. Velocity performance

The velocity curve varies between extremums of velocity (maxima, minima) which specifies the number of strokes. In fact, the effectiveness of pen velocity reconstruction from an offline image is to give sense and dynamic information to offline handwriting. Hence, we become able to segment an image into a primitive line based on the reconstructed pen tip velocity. In addition, we can obtain more features to improve the offline handwriting recognition rate. In this study, velocity reconstruction visually appears when plotting the character (see Fig. 11. b) where the points are not equidistant. In addition, the magnitude of the pen velocity (Fig. 11. d) shows the variation in acceleration as a function of time. As illustrated in Fig.11. we obtain an online signal; and based on the beta-elliptic model [27], there are two feature types, which are the dynamic and geometric profiles.

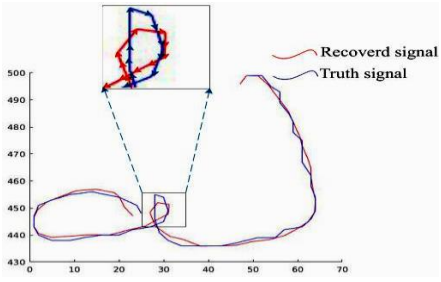


Fig. 7. Recovered signal and its counterpart of an Indian letter.

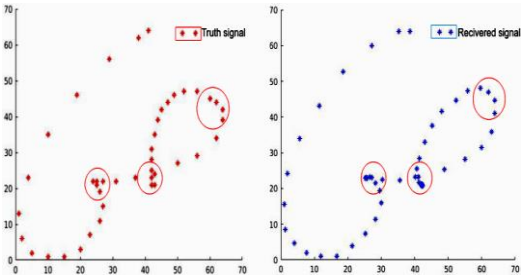


Fig. 8. Recovered pen velocity for Arabic letter /sa:d/.

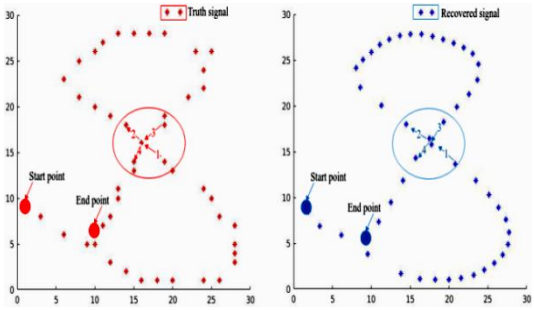


Fig. 9. Example shawing truth trajectory recovery for Digit "8".

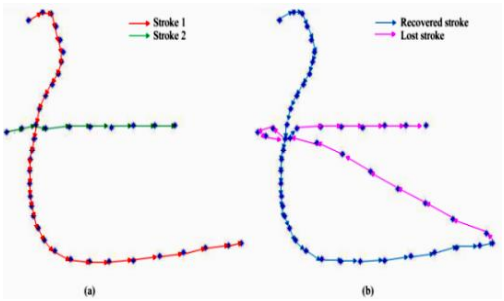


Fig. 10. Example showing up / down pen recovery problem for Latin character "t".

In the geometric profile, each beta stroke can be represented by an elliptic arc described by four geometric features: a, b, teta, and teta_p, where a and b are the half and small dimensions of the elliptic arc and teta and teta_p are respectively the angle of the ellipse and the tangent inclination. Those profiles are more detailed in [27].

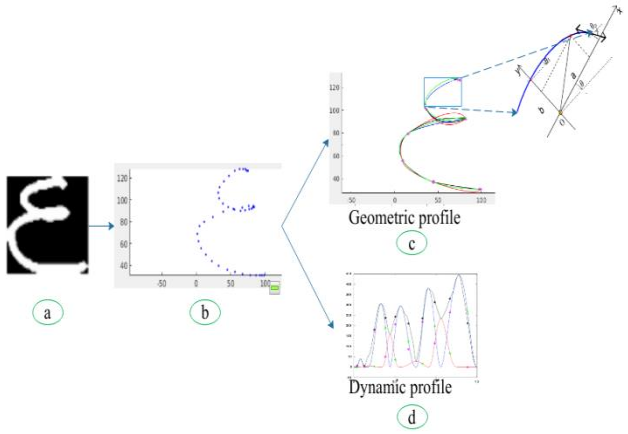


Fig. 11. Velocity reconstruction from offline handwriting. (a) Scanned image. (b) Recovery Cartesian coordinates x(t), y(t). (c) Geometric profile. (d) Magnitude of pen acceleration as a function of time.

5. Conclusion

In this study, we have introduced a novel framework based on a Seq2Seq model to recover the temporal order and pen velocity of multilingual handwriting characters. We have proved the importance of the attention model to focus on the local state while recovering online trajectories. The framework is an end-to-end system based on a CNN to extract features and an encoder-decoder BGRU with attention model to generate a signal with temporal order and velocity information. Consequently, we have achieved a higher recognition rate compared to other existing state-of-the-art models. Among the challenges that could be addressed in the future is to better recover the dynamic information from words, sentences and a complicated signature taking into consideration the pen up / down information. Thus, the use of meta learning will be required in this context, especially when we need to generalize the recovery model to unseen offline handwriting proficiently. We will also deploy a dependent bidirectional recurrent NN as it solves the Seq2Seq erroneous prediction problem, potentially improving the accuracy results. Finally, other deep learning and reinforcement learning models will need explored [40], and a range of benchmark multi-lingual handwriting character databases developed for comparative evaluation with other state-of-the-art (e.g. multi-task [41] and multi-model deep learning [42]) approaches.

Acknowledgment

The research leading to these results has received funding from the Ministry of Higher Education and Scientific Research of Tunisia under the grant agreement number LR11ES48.

References

[1] R. Arora, A. Basu, P. Mianjy,A. Mukherjee, "Understanding deep neural networks with rectified linear units," arXiv preprint arXiv:1611.01491, 2016.

[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.

[3] G. Boccignone, A. Chianese, L. P. Cordella, A. Marcelli, "Recovering dynamic information from static handwriting," Pattern recognition, 26(3), pp 409-418, 1993.

[4] H. Boubaker, A. ElBaati, M. Kherallah, A. M. Alimi, H. Elabed, "Online Arabic handwriting modeling system based on the graphemes segmentation," In 2010 20th International Conference on Pattern Recognition, pp. 2061-2064, 2010.

[5] H. Bunke, R. Ammann, G. Kaufmann, T. M. Ha, M. Schenkel, R. Seiler, F. Eggimann, "Recovery of temporal information of cursively handwritten words for on-line recognition," In Document Analysis and Recognition, Proceedings of the Fourth International Conference on Vol. 2, pp. 931-935, 1997.

[6] C. Chung, K. Gulcehre, Cho, Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv:1412.3555, 2014.

[7] G. Crispo, M. Diaz, A. Marcelli, M. A. Ferrer, "Tracking the Ballistic Trajectory in Complex and Long Handwritten Signatures," In 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 351-356, 2018.

[8] T. Dhieb, W. Ouarda, H. Boubaker, A. M. Alimi, "Deep neural network for online writer identification using beta-elliptic model," In Neural Networks (IJCNN), International Joint Conference, pp. 1863-1870, 2016.

[9] M. Dinh, H. J. Yang, G. S. Lee, S. H. Kim, L. N. Do, "Recovery of drawing order from multi-stroke English handwritten images based on graph models and ambiguous zone analysis," Expert Systems with Applications, 64, 352-364, 2016.

[10] A. ElBaati, A. M. Alimi, M. Charfi, A. Ennaji, "Recovery of temporal information from off-line arabic handwritten," In aiccsa, pp. 127-vii, 2005.

[11] A. Elbaati, H. Boubaker, M. Kherallah, A. Ennaji, H. El Abed, A. M. Alimi, "Arabic handwriting recognition using restored stroke chronology," In International Conference on Document Analysis and Recognition ICDAR'09, pp. 411-415, 2009.

[12] A. Elbaati, M. Kherallah, A. Ennaji, A. M. Alimi, "Temporal order recovery of the scanned handwriting," In International Conference on Document Analysis and Recognition ICDAR'09, pp. 1116-1120, 2009.

[13] Y. Hamdi, A. Chaabouni, H. Boubaker, A. M. Alimi, "Hybrid Neural Network and Genetic Algorithm for off-Lexicon Online Arabic Handwriting Recognition," In International Conference on Hybrid Intelligent Systems Springer, Cham, pp. 431-441, 2016.

[14] A. Hassaïne, S. Al Maadeed, A. Bouridane, "Icdar 2013 competition on handwriting stroke recovery from offline data," In International Conference on Document Analysis and Recognition (ICDAR), pp. 1412-1416, 2013.

[15] N. Chouikhi, B. Ammar, A. Hussain, A. M. Alimi, , " Bi-level multi-objective evolution of a Multi-Layered Echo-State Network Autoencoder for data representations," Neurocomputing, 341, 195-211, 2019.

[16] S. Hochreiter, J. Schmidhuber, "Long short-term memory," Neural computation, 9(8), 1735-1780, 1997.

[17] Y. Su, C. C. J. Kuo, "On extended long short-term memory and dependent bidirectional recurrent neural network," Neurocomputing, 356, 151-161, 2019.

[18] R. Jozefowicz, W. Zaremba, I. Sutskever, "An empirical exploration of recurrent network architectures," In Proceedings of the 32nd International Conference on Machine Learning (ICML-15) pp.2342-2350, 2015.

[19] L. Kang12, J. I. Toledo, P. Riba, M. Villegas, A. Fornés, M. Rusinol, "Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition," 40th German Conference on Pattern Recognition (GCPR), 2018.

[20] G. Kanojia, S. Raman, "DeepImSeq: Deep image sequencing for unsynchronized cameras," *Pattern Recognition Letters, 117*, 9-15, 2019.

[21] V. A. Kha, H. H. Kha, M. Blumenstein, "Extraction of Dynamic Trajectory on Multi-Stroke Static Handwriting Images Using Loop Analysis and Skeletal Graph Model," REV Journal on Electronics and Communications, 6(1-2), 2016.

[22] M. Kherallah, A. Elbaati, H. E. Abed, A. M. Alimi, "The on/off (LMCA) dual Arabic handwriting database," In 11th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2008.

[23] A. K. Bhunia, A. Bhowmick, A. K. Bhunia, A. Konwer, P. Banerjee, P. P. Roy, U. Pal, "Handwriting Trajectory Recovery using End-to-End Deep Encoder-Decoder Network," In 24th International Conference on Pattern Recognition (ICPR), pp. 3639-3644, 2018.

[24] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, v. 86, pp. 2278-2324, 1998.

[25] Z. Noubigh, M. Kherallah, "A survey on handwriting recognition based on the trajectory recovery technique," In Arabic Script Analysis and Recognition (ASAR), pp. 69-73, 2017.

[26] D. Phan, I. S. Na, S. H. Kim, G. S. Lee, H. J. Yang, "Triangulation Based Skeletonization and Trajectory Recovery for Handwritten Character Patterns," Ksii Transactions on Internet & Information Systems, 9(1), 2015.

[27] Y. Hamdi, H. Boubaker, T. Dhieb, A. Elbaati, A. M. Alimi, "Hybrid DBLSTM-SVM based Beta-elliptic-CNN Models for Online Arabic Characters Recognition," In International Conference on Document Analysis and Recognition ICDAR'19, pp. 545-550, 2019

[28] Y. Qiao, M. Nishiara, M. Yasuhara, "A framework toward restoration of writing order from single-stroked handwriting image," IEEE transactions on pattern analysis and machine intelligence, 28(11), 1724-1737, 2006.

[29] B. Rabhi, A. Elbaati, Y. Hamdi, A. M. Alimi, "Handwriting Recognition Based On Temporal Order Restored By The End-To-End System," In International Conference on Document Analysis and Recognition ICDAR'19, pp. 1231-1236, 2019.

[30] B. Rabhi, H. Dhahri, A. M. Alimi, "Grey Wolf Optimizer for Training Elman Neural Network," In International Conference on Hybrid Intelligent Systems, Springer Cham, pp. 380-390, 2016.

[31] M. Rosca, T. Breuel, "Sequence-to-Sequence neural network models for transliteration," arXiv preprint arXiv:1610.09565, 2016.

[32] L. Rousseau, E. Anquetil, J. Camillerapp, "Recovery of a drawing order from off-line isolated letters dedicated to on-line recognition," In International Conference on Document Analysis and Recognition, ICDAR, pp. 1121-1125, 2005.

[33] K. Simonyan, A. Zisserman," Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556 [cs], 1409.1556, September 2014.

[34] T. Steinherz, D. Doermann, E. Rivlin, N. Intrator, "Offline loop investigation for handwriting analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(2), 193-209, 2009.

[35] I. Sutskever, O. Vinyals, Q. V. Le, "Sequence to sequence learning with neural networks," In Advances in neural information processing systems, pp. 3104-3112, 2014.

[36] C. Viard-Gaudin, P.M. Lallican, S. Knerr, P. Binter, "The ireste on/off (ironoff) dual handwriting database," In International Conference on Document Analysis and Recognition, ICDAR'99 (Cat. No. PR00318) pp. 455-458, IEEE, 1999.

[37] H. Yu, J. Wang, Z. Huang, Y. Yang, W. Xu, "Video paragraph captioning using hierarchical recurrent neural networks," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4584-4593, 2016.

[38] J. Weston, S. Chopra, A. Bordes, "Memory networks," arXiv preprint arXiv:1410.3916, 2014.

[39] HE, Kaiming, Zhang, Xiangyu, REN, Shaoqing, et al. Deep residual learning for image recognition. In : Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.

[40] M. Mahmud, M.S Kaiser, A. Hussain, S. Vassanelli, "Applications of Deep Learning and Reinforcement Learning to Biological Data," IEEE Transactions in Neural Networks and Learning Systems, 29(6):2063-2079, 2018.

[41] F. Xiong, B. Sun, X. Yang, H. Qiao, K. Huang, A. Hussain, Z. Liu, "Guided Policy Search for Sequential Multitask Learning," IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49(1): 216-226, 2019.

[42] C. Ieracitano, N. Mammone, A. Hussain, F.C Morabito, "A novel multi-modal machine learning based approach for automatic classification of EEG recordings in dementia," Elsevier Neural Networks, Vol. 123:176-190, 2020.