Temporal Order and Pen Velocity Recovery for Character Handwriting Based on Sequence to Sequence Gated Recurrent Unit Model

SUBMISSION DATE / POSTED DATE

09-10-2020 / 16-10-2020

CITATION

Rabhi, Besma; Elbaati, Abdelkarim; Boubaker, Houcine; Alimi, Adel M. (2020): Temporal Order and Pen Velocity Recovery for Character Handwriting Based on Sequence to Sequence Gated Recurrent Unit Model. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.13072193.v1

DOI

10.36227/techrxiv.13072193.v1

Pattern Recognition Letters
journal homepage: www.elsevier.com

Temporal Order and Pen Velocity Recovery for Character Handwriting Based on Sequence to Sequence Gated Recurrent Unit Model

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

[a]*University of Sfax, National Engineering School of Sfax, REGIM-Lab.: REsearch Groups in Intelligent Machines, LR11ES48, 3038 Sfax, Tunisia*

ABSTRACT

The online signal is rich in dynamic features such as trajectory chronology, velocity, pressure and pen up/down. Their offline counterpart consists of a set of pixels. Thus, the online handwriting recognition accuracy is generally better than the offline one. In this paper, we propose an original framework for recovering temporal order and pen velocity from offline handwriting. Our framework is based on sequence to sequence Gated Recurrent Unit (Seq2Seq GRU) model. The proposed system consists in extracting a hidden representation from an image using Convolutional Neural Network (CNN) and Bidirectional GRU (BGRU), and decoding the encoded vectors to generate dynamic information using BGRU. We validate our framework by an online recognition system applied on Latin, Arabic and Indian On/Off dual handwriting character database. To prove the performance of the proposed system, we achieve a low error rate of point coordinates and a high accuracy rate of the LSTM recognition system.

Keywords:Temporal order recovery; pen velocity reconstruction; Deep Learning;BGRU;Sequence-to-Sequence 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 the handwriting is captured using different acquisition techniques, it makes* rise of two handwriting categories: an online type and an offline type. In the case of online handwriting category, it may require special digital devices and it represents numeric data presented as a succession of points ordered in time. Consequently, this mono-dimensional signal is noted as dynamic features; the temporal order, the pen velocity, pressure and the pen up/down. In contrast, the offline handwriting requires a camera or a scanner to capture handwriting from the paper. Therefore, the 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, consequently, 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 improve the recognition accuracy. In [29], authors demonstrate the effectiveness of the velocity in the recognition task applied on Arabic handwriting character. They obtained 98.8% for re-sampled online signal against 95.8% for 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] demonstrate the effectiveness of the online system compared to the offline one, using the recognition rates as evaluation metric. They obtained 96% for online digits against 90% for offline images. To exploit the advantages of offline and online treatments, researchers have presented many methods to recover the temporal order from static handwriting image. 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 has an effect on the next step. Moreover, these last methods suffer from the problem of assumption because direction will be different according to the language of the writing. In [25], authors affirm that trajectory chronology proves to be more promising for reconstruction, but there is 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. Recently, Memory Recurrent Networks [38] with the potential to treat long term sequential tasks, realize great success. Among these investigations, image caption [37] has achieved the level of translating images into text. Motivated by this, we assume that seq2seq models have a great potential to become the newly state-of-the-art for handwriting recovery problem. Seq2seq models correspond to an encoder-decoder model. Our framework contains a) the Convolutional Neural

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

Network (CNN) to extract the lower level features, b) the Bidirectional GRU (BGRU) to encode the last extracted features to a single vector, c) the BGRU to decode the encoded features into ordered coordinates. Actually, Seq2Seq with GRU NN is applied in different contexts; handwritten word recognition [19] and machine translation [31]. For the best of our knowledge we are the first to implement Seq2Seq-BGRU model for temporal order and pen velocity recovery. The major contributions of this work are displayed below.

- We started by investigating a novel Seq2Seq model based on BGRU NN to predict the dynamic information from a static handwriting image.
- We combined CNN and BGRU to extract features from images.
- We recovered, for the first time, the pen velocity in addition to the temporal order.
- This End-To-End system is able 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. Then, section 5 provides the implications of the study and the conclusion.

2. Related works

A set of works quoted in the literature recover the temporal trajectory order according to one category; contour or skeleton approach. The contour technique [10, 34] suffers from high computational time. For example, in [34], the authors are interested in loop analysis. They process different models for loop types and perform a thorough loop contour analysis. Nevertheless, the effectiveness of the proposed investigation on loops is not clear enough as they do not show any practical results of handwriting recognition and the valuation time is high. On the other hand, the use of skeleton approach gives good results and a more rapid response compared to contour method [9, 12, 21, 28, 32]. Based on Edge Continuity Relation (ECR) [28], the authors propose 3 main steps. First, they identify different ECRs at each node. In the case of node of degree four, they use the Neural Network, if not the case, they use some assumptions. Second, they select double-traced lines using the maximum weighted matching. Their last step is to find the smoothest possible path to go through all the curves of the handwriting graphic model. Based on the optimal Euler path, they select the smoothest one. However, their work is applied on single stroke for mono language. In [32], the authors use the handwriting knowledge to propose the possible start/end points. Afterward, different paths are produced and the best is chosen. Their approach is applied on multi stroke letters. To demonstrate the performance of their approach, they present a good recognition rate. Even so, they use the assumptions based on the Latin language only. In addition to the presence of contour and skeleton categories, surrounding areas of handwriting recovery can be divided into two groups: local and global search method. The local tracing method goal is 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 is that the design of heuristic rules applied for different handwriting styles is difficult. This limitation can be overcome by using the global graph technique. It aims to create a graph model of the input skeleton images and then use a search technique to find an optimal path through the text [5, 12, 21, 28, 32]. The drawback of global method is that the computational time is high and it depends on the complexity of the algorithms used as search technique and also there are some cases which are hardly treated. For example, Phan et al. [26], use greedy algorithm for searching about the optimal path in a global model. Their work is based on limited assumptions about start/end points, ambiguous zone and double trace segments, which give rise to a difficult decision for obtaining the right trajectory. In [9], the authors consider that start point detection and skeleton separation is a hard task. In addition, there is a higher complexity of searching the smoothest path at junction zone. Based on skeleton graph, they separate touching characters and crossing strokes. The optimal path is fixed by a Greedy algorithm. Their model is still sensitive to the processed language and they have the common problem of being slow and complex [19, 28, 32]. However, it is not clear whether the use of the local method can achieve a more effective performance compared to the global method or on the contrary. Both of them suffer from some problems. Thus some existing works have combined these two tracing methods together [12], in which they optimize the number of possibilities by adding some local features such as: curvature and inclination angle. It cannot be denied that previous works got perfect performances particularly in Latin language. But most of them are weakened by some language like Arabic handwriting corpus, which makes them have many versions for each language [12]. Moreover, the decade old problem of handwriting recovery is based on finding out the correct terminal points (start/end points), junction points and the main direction in the detected ambiguous zone. In addition, Rousseau et al. [32] demonstrate that each step of recovery procedure can affect the results of recognition rate. Thus, in our opinion, these challenging parts are complicated. The early works [23, 29] address the use of an End-To-End system for handwriting recovery. In [29], authors use VGG-LSTM to extract features from image and BLSTM is used as a decoder model. Their system is the most closely related work. In this paper, though, we refer to our previous work [29] where the focus is different. We use CNN-BGRU to extract features from images, instead of VGG-LSTM. In the task of recovering temporal order from offline handwriting it is more challenging to produce a human-like velocity. Authors in [29] recover an online signal with equidistant points. They use the 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. Since our framework is based on CNN-GRU architecture for recovering the temporal order and the pen velocity, it is called CG-Velocity.

3. END-TO-END Recovery Framework

In this section, we clarify the main architecture of the proposed framework. We employed the Seq2Seq [35] model to transform a sequence of offline handwriting character into a sequence of its 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:

$$P[pc|fp] \quad (1)$$

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

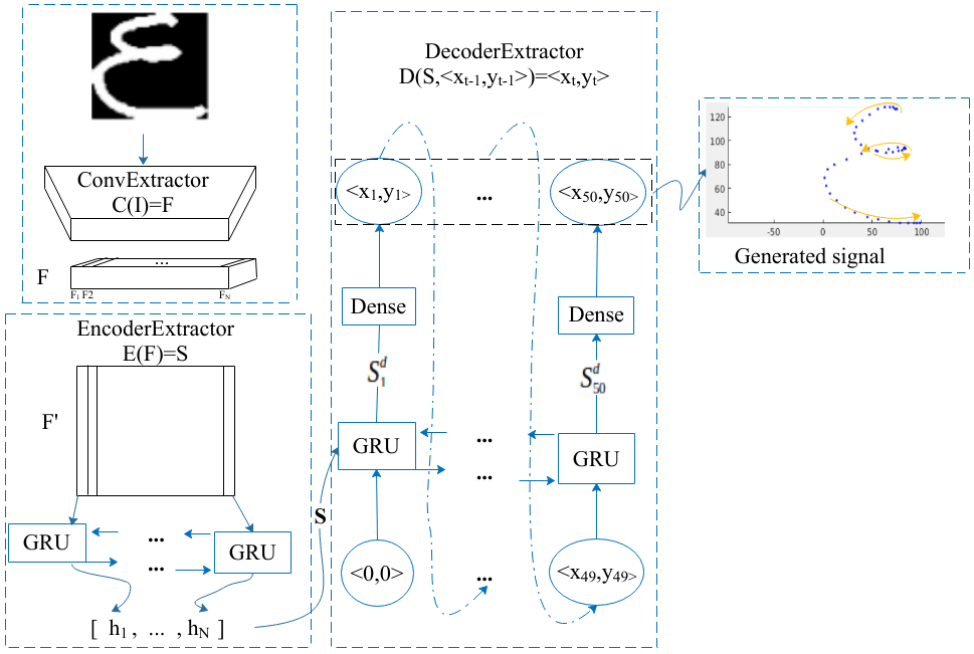


Fig. 1. Architecture of the Seq2Seq BGRU model. For sake of simplicity, one BGRU layer is shown.

The EncoderExtractor function E() is a multilayer BGRU model that accepts the sequence of features F and extracts the last state S. The main role of the encoder is to build some hidden representation of the input sequence to get the context vector. In literature, the encoder part can be CNN followed by a dense layer [33] (CNN-Dense), or followed by an LSTM/GRU (CNN-GRU). The comparison between them is presented in section 4. Finally, the DecoderExtractor D() function is a multilayer BGRU model. It receives the last encoder state as its initial state. It generates pixel by pixel coordinates combining the hidden state S and the previous predicted coordinates. To summarize, the following equations present the recent processes:

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

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

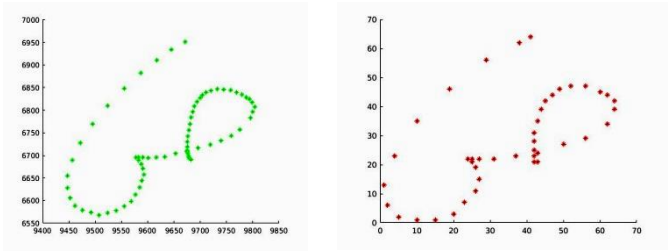
$$(x_t,y_t)=D(S, <x_{t-1},y_{t-1}>) \tag{4}$$

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 b) its counterpart online signal. Generally, the first step in handwriting recovery is the 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 signal 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 Algorithm of normalization step). In actual fact, the online handwriting is a series of no 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. For the best of our knowledge, authors did not reconstruct the pen velocity when solving the handwriting recovery problem. However, in [12, 29], authors use the re-sampling step to add the velocity to the recovered signal. But what about generating a significant signal with temporal order and pen velocity in the same time?

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 signal is used as input to the Algorithm of normalization step. Fig. 2 (b) represents the output signal after the 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 an online signal of the Arabic letter "sad". Green color indicates the original online signal (a), red color represents the normalized signal (b).

The GRU and the LSTM Network can be adapted to different sizes (with 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, the distance between two points will depends on the natural speed of the pen but also on the total length of the character. In consequence, the strength of the following algorithm is to solve this issue and normalize the online signal with fixed point number.

Algorithm: Normalization step with fixed point number

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

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

1: Calculate the curvilinear length Cl of the original trajectory
2: Calculation of the trajectory average speeds obtained in the original and expected resampling respectively :
Vm_I = Cl / (Δt . N_I); Vm_O = Cl / (Δt . N_O) ;
3: Calculate the curvilinear abscissa x_curved_I(j) of each N_I (the distance from the start point to the asked point)
4: Initialize the current point of the resampled trajectory: M_O (1,:) = M_I (1,:)
5: Initialize the curvilinear abscissa of the current point of the

resampled trajectory: x_curved_O(1) = 0
6: Initialize the instantaneous speed / Average speed proportionality factor at the current point : p_f = 1; j=2
7: **For** : i = 2 **to** (N_O)
7.1: calculate the curvilinear abscissa of the new point
x_curved_O(i) = x_curved_O(i-1)+ (Vm_O . P_f. Δt)
7.2 : Position the new current point M_O (i) between two successive points M_I (j) and M_I (j + 1) of the 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 the 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 the new point M_O (i + 1) to the matrix of points of the re-sampled trajectory
end For
end

3.2. Convolutional extractor

Deep CNN [24] have been used in different tasks [19, 23, 37]. Given CNN's success in providing a solid representation of features, we use CNN 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 rectangle region. Those features describe the image region. The details of CNN configuration are described in section 4. The input image (64*64) is transformed to CNN-feature of size (Bachsize, N, D). Specifically, Bachsize is fixed to 32, N and D are the length and the depth of the CNN entity, respectively. As shown in Fig. 1 the CNN output is noticed by F = (F_1, F_2, ..., F_N) and $F_i \in R^D$ (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 actually processes the input to produce a single hidden representation of all the input information. Among the varied types of Neural Nets, RNN is able to forecast the most accurate results [15, 30]. In fact, the famous problem of simple RNN is the vanishing gradient problem, which LSTM [16] and GRU [6] can alleviate. According to previous studies [18], there are no conclusions that can be drawn about the winner LSTM or GRU. Both RNNs achieve similar results in many tasks. Even so, GRU can be better in terms of time thanks to parameter size. In addition, according to [17], GRU could be a better choice for modeling the temporal information compared to LSTM, that's why 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 the updated gate, the reset gate and the candidate activation values, respectively. σ is the sigmoid function. F_t is the input which is the extracted CNN features. W_{fg}, U_{hg}, W_{fr}, U_{hr}, W_{fh} *and* U_{rh} represent different weights. To obtain more information and better representation, BGRU with multiple layers is chosen as the most efficient encoder type adapted to our problem. Precisely, after many experiments of different 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 shown in Fig. 1 ConvExtractor generates an intermediate feature F which is reshaped to F' map (F'_1, F'_2, …, F'_N) with two dimensional features. Those features are used as input to the first layer of the BGRU encoder.The remaining layers use as input its previous hidden state. For each time step, the output of the encoder is $h_t \in$ H calculated by the Eq.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 DecoderExtractor is to generate the predicted coordinate sequences, as shown in Eq. 9:

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

Here $P_1, \dots, P_i = [<x_1,y_1>,\dots,<x_i,y_i>]$ where *i* is the number of the desired points. F is the set of features and $F_N \in R^D$ where *N* and *D* are the length and depth of CNN, respectively. According to the entire feature sequence (F_1,..., F_N), the decoder estimates a set of points ($P_1,\dots,P_i$). *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 input the last state of the encoder S and the first coordinate <0,0>. Then, the remaining layers receive as 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 Eq. 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 and S_{i-1}^d is its previous state, g is a GRU function. The generated coordinates are calculated by the Eq. 11 below:

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

Here, we apply a linear activation function (*dense*). *U* and *b* are the weights and bias, respectively. 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 predicted point coordinates values are updated according to the loss *L1* Eq. 12 :

$$L_1=\frac{1}{N}\sum_{t=1}^{N}|P(t)-P'(t)| \quad (12)$$

Let the ground truth vector be $P'= [<x'_1,y'_1>,\dots,<x'_i,y'_i>]$ where *i* is the number of points. We calculate the loss *L1* using the predicted vector P. The training continued until the loss *L1* converges, the model is saved. The test step uses the offline handwriting image as input and, based on the obtained model, the

framework can generate a set of point coordinates representing a human-like writing.

3.5. Velocity reconstruction performance

With the rise of Deep Learning, handwriting recovery can be handled so that it can produce a new feature such as the pen velocity. The velocity curve varied 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 the 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, the velocity reconstruction appears visually when plotting the character (see Fig. 3. b) where the points are not equidistant. In addition, the magnitude of the pen velocity (Fig. 3. d) shows the variation of the acceleration as a function of time.

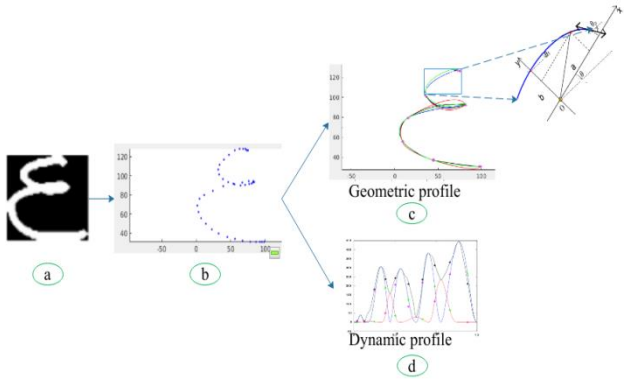


Fig. 3. Velocity reconstruction from offline handwriting. (a) Scanned image. (b) Recovery Cartesian coordinates x(t), y(t). (c) Geometric profile. (d) Magnitude of the pen acceleration as a function of time.

As shown in Fig.3. we obtain an online signal and based on the Beta-Elliptic-Model BEM [27] there are two feature types, which are the dynamic and the geometric profiles. In the geometric profile, each beta stroke can be represented by an elliptic arc described by four geometric features a, b, teta, teta_p where a and b are the half and the small dimensions of the elliptic arc. teta and teta_p are the angle of the ellipse and tangent inclination respectively. Those profiles are more detailed in [27].

4. Implementation and evaluation results

4.1. Ablation studies

The first study consisted in selecting the best CNN configuration. We tried different settings to train the framework on the IRONOFF [36] digit dataset. Here, we conserve the same encoder/decoder types (BGRU-3Layers-512Units) for all CNN models. As shown in Table 1, there are three variations of CNN model. We changed the number of layers and filters. CNN2 is the best combination which achieves the highest 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 characteristic from the previous convolution layers' output. Each layer is preceded by an activation function RELU (Rectified Linear Unit) [1]. Batch normalization is employed after the third and the last conv layer thereafter.

Table 1. CNN configuration of our architecture

	Parameter values for each layer	Training Accuracy
Conv. filters	64 – 128 – 256 – 256 – 512 - 512	80.1%
MaxPooling	(2,2) – (2,2) - No -(1,2) –(2,1)-No	
Conv. filters	64 – 128 – 256 – 256 – 256 – 256-256-512	**96.3%**
MaxPooling	(2,2) – (2,1) - No -(2,1) –(2,1)-No-(2,1)-No	
Conv. filters	50 –100–150–200–250 – 300-350-400-512	85.6%
MaxPooling	No – (2,2) -No -(2,2) –No-(1,2)-No-(2,1)-No	

In order to demonstrate the use of CNN-GRU as an encoder, we compare different models in terms of training time and efficiency, with one Layer and 256 units for all encoder types. The CNN-GRU/LSTM encoder has been used in similar handwriting recovery context [29, 23]. Table 2 shows the strength of CNN-GRU on IRONOFF digit database with batch size is 32, learning rate is 0.0001 and iteration is 400. CNN-Dense has negative impacts on the training performance and CNN-LSTM also degrades the accuracy. To select the best decoder type we train the framework with CNN2 as convolutional extractor and BGRU-3Layers (512) as encoder extractor. Table 2 demonstrates the performance of the decoder BGRU with 3 layers and 512 units in term of training accuracy.

Table 2. Encoder selection

		Training rate	Time/step
Encoder-1Layer	**CNN-Dense**	89.7%	0.3 s
	CNN-GRU	**94.0%**	**0.4 s**
	CNN-LSTM	93.2%	1 s
Decoder-512Units	**BGRU-1Layer**	94.1%	0.4s
	BGRU-2Layer	95.6%	0.4s
	BGRU-3Layer	**96.3%**	0.5s

4.2. Datasets and settings

We test the proposed framework CG-Velocity on Arabic, Latin and Indian corpus. Specifically, for Arabic, we use the dual on/off Arabic dataset LMCA [22]. It contains 28 letters, which are joined or isolated (we chose the isolated form). In the case of Latin data, we adopt the dual on/off Latin IRONOFF dataset. We use the isolated letters (upper and lower cases) and digits, sorted into 26 and 10 classes respectively. For Indian language, we use Telugu[1] dataset. It contains 116 Telugu characters. All these datasets are used without taking into account the pen up/down. We created an additional patterns using data augmentation strategy based on distorted samples (changing angle inclination, smoothing, and baselines). The obtained signals were 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 such that, if the current point is foreground, all its neighbors are set to the foreground. Table 3. Represents the number of samples used for training, testing and validation. 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 Root Mean Square Error (RMSE) and Euclidean Distance (ED), but also recognized by an online recognition system. Our framework is trained over one Nvidia GT 650M with GPU in Tensorflow platform. The training batch has 32 image-signal pairs. For parameters update, we chose Adam Stochastic with 10^{-3} of learning rate. We recuperate the checkpoint model every 700 iterations. Training time lasted for 16 hours and test step took around 7 minutes for each 20 samples.

Table 3. Dataset details

Scripts	Training	Testing	Validation
LMCA	28000	8400	5600
IRONOFF Lower-case	26000	7800	5200
IRONOFF upper-case	26000	7800	5200
Digits	10000	3000	2000
Telugu	58000	17400	11600

4.3. Evaluation metrics and models description

We state the effectiveness of our proposed framework on three evaluation metrics: Root Mean Square Error (RMSE), Euclidean Distance (ED) and an online recognition system based on LSTM NN. RMSE and ED are chosen as evaluation criteria for distance parsing. 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

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

same environment conditions. Specifically, Elbaati [12] approach is based on the graph model to represent an image as a set of segments type. Genetic Algorithm is used to find the smoothest path across those segments. For Ayan [23] system, authors present an End-To-End model based on CNN and BLSTM. For Rabhi [29] framework, authors present a Seq2Seq model based on VGG-16 and BLSTM. They used the re-sampling step after obtaining the recovered signal. Moreover, we choose other already published results for comparison such as Rousseau [32] and Qiao [28] which are articulated under the context of handwriting recovery. They use the methods described previously in section 2.

4.3.1. Root Mean Square Error

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

$$\boldsymbol{RMSE} = \frac{\mathbf{1}}{\mathbf{2}\boldsymbol{L}}\left(\sum_{i=1}^{n}\sum_{t=1}^{li}(\boldsymbol{x_t} - \boldsymbol{x'_t})^2 + \sum_{i=1}^{n}\sum_{t=1}^{li}(\boldsymbol{y_t} - \boldsymbol{y'_t})^2\right) \quad (13)$$

Where n is the number of samples and l_i is the number of points in each sample. x_t and y_t are the coordinates of the online signal and x'_t , y'_t represent the recovered coordinates. 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. It can be seen that our proposed CG-Velocity achieves the best results. It outperforms Elbaati [12] approach, Ayan [23] system and Rabhi [29] framework in 16, 0.3 and 0.1 absolute error points, respectively, when testing the digits data.

Table 4. Results of RMSE rate compared to located systems

	IRONOFF Lower-case	IRONOFF Upper-case	IRONOFF Digits	LMCA	Telugu
Elbaati[12]	20.1	19.0	**18.0**	32.9	20.1
Ayan[23]	3.6	2.9	**2.3**	2.5	3.2
Rabhi[29]	3.5	2.3	**2.1**	2.2	3.1
Proposed	**3.0**	**2.5**	**2.0**	**2.2**	**2.7**

4.3.2. Euclidean Distance

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

$$ED = \sqrt{\left(x_p - x'_q\right)^2 + \left(y_p - y'_q\right)^2} \quad (14)$$

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 better method is the one that achieves the lower values. The results are shown in Table 5. It can be affirmed that our proposed framework achieves the best performance. It surpasses Elbaati [12] approach, Ayan [23] system and Rabhi [29] in 30.9, 0.2 and 0.1 absolute error points, respectively, when testing the Arabic data.

Table 5. Results of ED error rate compared to located systems

	IRONOFF Lower-case	IRONOFF Upper-case	IRONOFF Digits	LMCA	Telugu
Elbaati[12]	43.0	38.0	35.0	52.0	38.4
Ayan[23]	32.2	23.9	20.3	21.3	24.2
Rabhi[29]	32.1	23.8	20.1	21.2	23.7
Proposed	**32.0**	**23.2**	**20.0**	**21.1**	**23.4**

4.3.3. Handwriting recognition

Fig. 4 shows our recovered signal with a small deviation compared to its counterpart ground truth signal. In the same time the temporal order is suitable for the online one. In consequence, we agree that the evaluator metrics RMSE and even ED are brutal when evaluating a predicted signal with a small deviation, even if the temporal order is compatible with its ground truth, but the accuracy result could be unexpected. After this interpretation, we used an online LSTM recognition system [29], based on beta elliptic and grapheme segmentation method [4] that requires the existence of the temporal order and the velocity to produce a reasonable result. We extract 118 characteristics from the original online signal to train the network. Then, we test the network with our reconstructed signal.

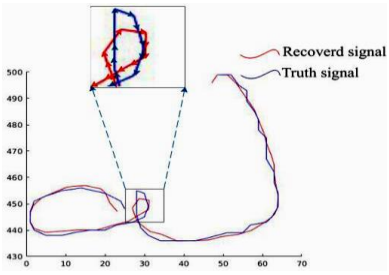


Fig. 4. Recovered signal and its counterpart of an Indian letter.

- Results of handwriting recognition rate compared to located systems.

As proved in Table 6, the recognition rate, whatever the database, is higher than other methods. We gain a high margin effectiveness improvement over Elbaati [12]. Their model is sensitive to noise (65.2% for digits). Based on assumptions, Elbaati model achieves a lower rate when treating the Arabic language.

Table 6. Results of handwriting recognition rate compared to located systems

	Lower-case	Upper-case	Digits	LMCA	Telugu
Elbaati[12]	48.3%	49.2%	65.2%	32.0%	48.5%
Ayan[23]	88.9%	90.4%	89.3%	91.9%	89.2%
Rabhi[29]	92.8%	94.1%	94.5%	98.8%	92.9%
CG-Velocity (ours)	**93.1%**	**94.4%**	**95.6%**	**98.9%**	**93.2%**
Ground_truth	**92.6%**	**94.2%**	**94.4%**	**98.8%**	**92.9%**
CNN_GRU without velocity	**92.0 %**	**94.0%**	**94.0%**	**97.0%**	**92.7%**

Besides, Ayan [23] system is the closest method to ours in terms of applying an end-to-end model. Their higher accuracy rate (91.9%) was achieved when testing the LMCA dataset but it remains below our rate (98.9%) because the velocity aspect is not treated in their system. Regarding the lower accuracy achieved by Rabhi [29] framework, we can interpret that GRU NN proves its efficiency in terms of accuracy and time (0.7s/step) compared to LSTM (1s/step). To show the effectiveness of the pen velocity process proposed in this paper, we assess the model on another scheme without velocity: CNN_GRU without velocity, which is obtained after trained the framework with the offline handwriting and its corresponding online one (a set of equidistant points). By making this comparison, we find that using the pen velocity is very efficient for handwriting recognition parsing. Therefore, we can get better results compared to the ground truth signal, because the generated signal is a human like writing and the obtained signal is normalized (see Table 6). However, the effectiveness of the velocity is captured when the CNN_GRU without velocity (97%) is lower than ground truth rate (98.8%).

- Results of handwriting recognition rate compared to existing works.

As proved in Table 7, the author's method in [32] is based on a global graph model. However, they mention the problem of selecting the start/end points with hidden representation. Thus, they remove some letters that represent around 9% of data. They achieve (87%) less than ours (94.4%). In [28], authors use the Digits data and achieve a good accuracy 95% but still lower than ours (95.5%).

Table 7. Results of handwriting recognition rate compared to existing systems

	Alphabet	Digits
Qiao et al. [28]	-	95%
Rousseau et al. [32]	87%	-
Proposed	**94.4%**	**95.5%**

4.4. Figures Analysis

The proposed framework uses as input an offline image and generates an online signal with temporal order and pen velocity. Fig. 5 represents the pen velocity of our recovered signal and the truth signal. The acceleration decreases at the red circles zones as the ground truth signal. This makes us admit that the recovered signal respects the velocity as a human writing.

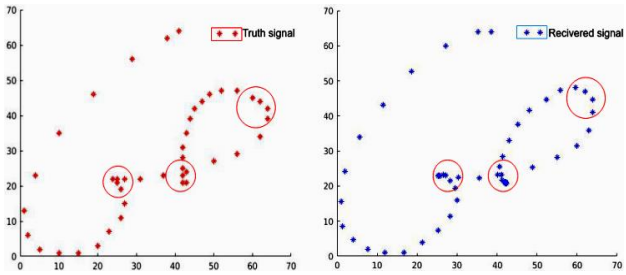


Fig. 5. Recovered pen velocity for the Arabic letter « sad ».

As indicated in Fig. 6 the recovered signal passes the loop in 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 truth signal.

Fig. 7 indicates some cases of the proposed 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 focused on the reconstruction of the pen velocity feature. Which are not yet treated neither with old systems nor with new ones.

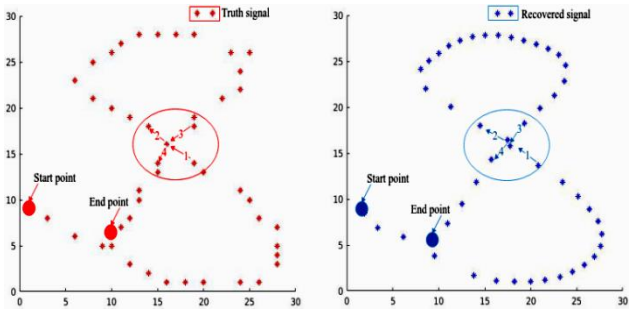


Fig. 6. Example shawing truth trajectory recovery for the Digit eight.

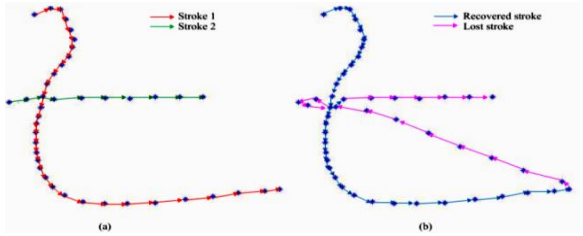


Fig. 7. Example showing up/down pen recovery problem for the Latin character "t".

5. Conclusion

In this study, we have proposed a novel framework based on sequence to sequence model to recover the temporal order and the pen velocity of a multilingual handwriting characters. The proposed framework is an End-To-End system based on a CNN to extract features and an Encoder-Decoder BGRU model to generate a signal with temporal order and velocity information as well. Consequently, we achieve a higher recognition rate compared to other existing 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 account the pen up/down information. Thus, the use of seq2seq model will be required in this context, especially the use of the attention model. It is also recommended to try out dependent bidirectional RNN (DBRNN) as it solves the Seq2Seq's erroneous prediction problem, hence, to improve the accuracy results.

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] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," In International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02), 107-116, 1998.

[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. N. Haque, M. A. Yousuf, R. Rana, “Image denoising and restoration with CNN-LSTM Encoder Decoder with Direct Attention,” arXiv preprint arXiv:1801.05141, 2018.

[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.

J. Weston, S. Chopra, A. Bordes, “Memory networks,” arXiv preprint arXiv:1410.3916, 2014.