On the Use of Attention Mechanism in a Seq2Seq based Approach for Off-line Handwritten Digit String Recognition

T Lupinski, A. Belaid, A Echi

On the Use of Attention Mechanism in a Seq2Seq based Approach for Off-line Handwritten Digit String Recognition

T. Lupinski and A. Belaïd
Université de Lorraine - LORIA
Nancy, France
ts.lupinski@gmail.com, abdel.belaid@loria.fr

A. Kacem Echi
Université de Tunis - ENSIT - LaTICE
Tunis, Tunisia
Email: afef.kacem@ensit.rnu.tn

***Abstract*—In this work, we investigate the use of the attention mechanism in deep learning for a better reading of handwritten digit strings in digitized images. The proposed recognition system built upon a CNN (Convolutional Neural Network) and two RNNs (Recurrent Neural Networks), acting as Encoder and Decoder and using the attention mechanism. We used a 1D mechanism for attention location with a "soft" alignment attention which has the peculiarity of having an easily calculable gradient and thus to integrate well with the network. Experimental results on data from ORAND-CAR A, ORAND-CAR B and CVL HDS databases compare favorably to other published methods.**

***Keywords*-end-to-end trainable system; sequence-to-sequence; attention mechanism; handwritten digit string; transfer learning;**

I. Introduction

The recognition of handwritten digit string is an important subject of research in the field of document analysis and recognition. It is motivated by the wide variety of potential applications, such as ZIP codes, bank checks, etc. The challenge is to recognize numeral strings of unknown length which are not neatly written. Although the recognition of isolated digits is now very successful, the recognition of digit strings remains an ongoing challenge in the field of handwriting recognition. In fact, the recognition of digits is considered easier as there are less possible symbols, but no language model can help in this task. The previously proposed recognition systems, achieving the best results, usually rely on a heavy set of heuristics and over-segmentation and show their limitations when faced with complex cases such as overlapping digits. Most of systems built to recognize unconstrained numerical strings suffer from the segmentation module which reads a string of digits and segments them into isolated characters. The problem is that we do not usually know the number of digits in the string and so the optimal boundary between them is unknown. Such a problem has been dealt with in different ways [5], [9], [14] and one way to approach it is to see it as a sequence-to-sequence problem with a sequence of image patches as input and a sequence of characters as output [2]. To solve sequence-to-sequence problems with different length from input to output, the key point is to find the alignment between them. Notice that for digit string recognition, the alignment is not straightforward. The problem of the alignment is generally handled either by posterior or prior probabilities. Based on posterior probabilities, we recourse to stochastic models like HMM (Hidden Markov Model) to extract the alignment from the probabilities outputed for each sub-part by looking for the most probable sequence of labels to fit the output [3]. In the same way and more recently, the CTC (Connectionist Temporal Classification) [4] has showed successes in many sequence labeling tasks with the strong ability of dealing with the problems where the alignment between the inputs and the target labels is unknown. In the prior probabilities based alignment, the attention mechanism is most often used in sequence-to-sequence models to learn many-to-one alignment [8]. Moreover, it had led to better results than HMM and CTC, for text translation [8], speech recognition [11] and word recognition [2]. Combining the attention mechanism with HMM or CTC might also lead to improvements [15]. For handwritten digit string recognition, the best results are achieved, using the CTC [6]. In this work, the objective is to explore the use of the attention-based approach for the specific task of handwritten digit string recognition. To support our opinion, we opted for an Encoder-Decoder paradigm, as in [1] and proposed a recognition system built upon a CNN (Convolutional Neural Network) and two RNNs (Recurrent Neural Networks) which are powerful neural network architectures used for modeling sequences. Here, the two RNNs act as Encoder and Decoder and use the attention mechanism which enables the system to focus on relevant parts of the input more than the irrelevant parts and consequently to enhance its recognition rate and speed up the processing of large volumes of data. A description of the proposed approach, along with the preliminary experimental results is presented here.

II. Proposed approach

The proposed approach is inspired from the sequence-to-sequence approach of Sutskever and al. [1] and motivated by its successful application on handwritten word recognition by Sueiras and al. [2]. As it will be explained later, our system is able to transform a variable-length sequence of

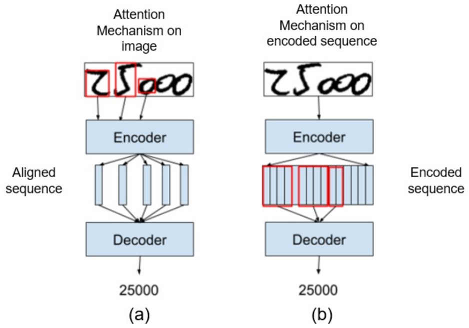


Figure 1. Attention mechanism location: (a) Before the RNN Encoder, (b) After the RNN Encoder.

pixel columns, extracted from the handwritten digit string image, into a variable-length sequence of digits to form a numerical string. As the typical sequence-sequence architecture consists of an encoder and a decoder RNN, our system is built with on three main components which are a CNN and two RNNs used as Encoder and Decoder respectively. A potential issue with this Encoder and Decoder approach is that the neural network needs to be able to compress all the necessary information of an input sequence into a fixed-length vector. This is very hard to do in practice, and this problem gets worse the longer the input sequence is. Such problem can be solved using the attention mechanism which allows the Decoder to attend to different parts of the source sentence at each step of the output generation. Thus, each Decoder output depends not just on the last decoder state, but on a weighted combination of all the input states. In the next subsections, the focus is on the use of the attention mechanism in our system: the best location, the used alignment, the energy and the context vector computing.

A. Attention Mechanism Location

Two possible locations of attention mechanism in our system:

- Before the RNN Encoder, the attention in made in 2D (see Figure 1(a)): This allows to play on the verticality of the image, by visualizing a 2D window of attention. It chooses the zones of the image to look at, move the attention to the right by following the digits one by one.
- After the RNN Encoder, the attention in made in 1D (see Figure 1(b)): It selects the parts of the sequence to keep for digit recognition. It singles a portion of the encoded sequence as a potential digit for the Decoder. We opted for this sequential attention which is the simplest.

B. Attention Mechanism Alignment

The use of the attention mechanism vary according to the context and there is no predefined model, but one can classify the mechanisms in several categories:

- “Hard” attention: The mechanism uses a predefined function, which can be driven, and which outputs a sub-part of the entries on which to work.
- “Soft” attention: The mechanism will calculate probabilities so that an area is interesting and weighted entries according to their interest, the output will be of the same size as the input but with its values weighted to bring out one or more areas.
- Feature-based: The attention mechanism will choose where to focus its attention based on what it looks at by detecting interesting areas.
- Location-based: The attention mechanism will choose where to pay attention depending on the position where it was looking before. Feature-based and Location-based are not exclusive and a mechanism can well use both.

For alignment, we had to learn it, since no rule helps us to give the alignment given images diversity. Here, we used a “soft” attention which has the peculiarity of having an easily calculable gradient and thus to integrate well with the network and to be trained at the same time as the latter contrary to the “hard” attention.

C. Attention Mechanism Energy Computing

To choose which part of the inputs to focus on, the mechanism needs to perform a score for each input, which we call energy. The most common ways to compute such energy are:

- Multiplication between each entry and the context vector: $e_i = h_i \times c_{t-1}$
- Weighted multiplication: $e_i = W \times h_i \times c_{t-1}$
- Weighted addition: $e_i = W \times h_i + U \times c_{t-1}$

A neural network of Perceptron type is often used to compute this energy by learning weights. For energy computing, we chose the additive form with $tanh$: a non linear activation function. Once computed, the energy is normalized with a Softmax function to have weights between 0 and 1.

D. Context Vector Computing

The context vector is the output of the attention mechanism. It is computed by weighting each entry by its associated energy and then summing these vectors. The context vector will be supplied to the Decoder for each time step. The number of time steps and the size of the output vector sequence are predefined. It is up to the Decoder to choose when to stop.

E. Retained Architecture

The global architecture is inspired from [7] and summarized in Table I. The proposed system consists of three stages as follows:

- Features Extraction: the CNN gets out a feature map, from the input image, and divides it into a sequence of vectors columns. The max pooling is repeated here to

Table I
NETWORK SETTINGS SUMMARY, FROM TOP TO BOTTOM.

Type	Configuration
Input	Input image, 256x32
Convolution	#maps: 64, k3x3, s1x1, p1x1, act:Relu
MaxPooling	k2x2,s1x1
Convolution	#maps: 128, k3x3, s1x1, p1x1, act:Relu
BatchNormalisation	-
MaxPooling	k1x2,s1x1
Convolution	#maps: 256, k3x3, s1x1, p1x1, act:Relu
BatchNormalisation	-
MaxPooling	k1x2,s1x1
Convolution	#maps: 512, k3x3, s1x1, p1x1, act:Relu
BatchNormalisation	-
MaxPooling	k1x2,s1x1
Convolution	#maps: 512, k3x3, s1x1, p1x1, act:Relu
BatchNormalisation	-
Flatten	Axis 2 & 3
LSTM	#hidden units:256
LSTM	#hidden units:256
Attention Cell	
LSTM	#hidden units:256
InnerProduct	#units:11
Softmax	-

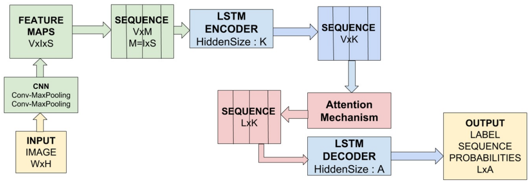


Figure 2. System overview.

reduce the size of the images by half, height or width alternately. Several sizes of filters are used to extract very fine features

- Encoder: an LSTM (Long Short Term Memory) which is an RNN architecture good at capturing long-term dependencies in the sequences and generally used when the learning problem is sequential. It takes as input the sequence of vectors and returns a sequence of vectors encoded, considering the previous and next context.
- Decoder: an LSTM with a mechanism of attention which at each stage of time will focus its attention on part of the coded sequence and give the probability that a digit is there. It can also predict the end of a number, but once predicted the other results are no longer taken into account.

Figure 2 shows an overview of the proposed system. At the output of the CNN, the feature maps of size ($V \times I \times S$) are subdivided into sequences ($V \times M$ where $M = I \times S$). Then, the first LSTM Encoder produces a sequence ($V \times K$). These sequences are considered by the attention mechanism which selects L sequences of size K. Theses sequences are then provided to the second decoder LSTM. At the end, a sequence of labels ($L \times A$) is produced. Hereafter more details about the architecture components are given.

1) Image input: . We normalized input images in height, in width and in value for channels. As previously mentioned, the images must have the same height, limited here to 32 pixels as a compromise between efficiency and computation costs. We then completed the images with 0 value to make them 256 pixels wide, allowing the use of any batch size what speed up the training. We finally resorted to histogram stretching so that each image has pixel values between 0 and 255, invert them to have “white on black” instead of “black on white” images and adjusted them so that the mean is equal to zero and the standard deviation equal to one. All this is done for convenience purposes.

2) CNN for Feature Extraction: Convolution network is a powerful tool for extracting features. We used a classical sequence of convolutional layers followed by max pooling layers, doubling the number of filters of convolutional layers up to 512 after each max pooling layer. The feature maps obtained after all these layers are flattened into one features map, then divided into a sequence of column features vectors, ordered from “left to right”. As in [6], we used a 2×2 pooling size for the first max pooling layer. For the next max pooling layers, we used a 1×2 pooling size to down sample just in height dimension, not in width. By this way, we kept the sequence size long enough to have more precision.

3) Encoder-Decoder: The Encoder takes the sequence of features vectors and transforms it into a sequence of encoded vectors. Then, the Decoder sequentially uses an attention mechanism to choose a part of this encoded sequence and decodes one character out of it, until the “End Of String” (EOS) token is outputted by the Decoder. The key difference between the Encoder and the Decoder is the order of the execution: as the attention mechanism needs the previous cell states of the last layer of the Decoder, the computation can only be done step by step instead of layer by layer. This model can handle images of any width as both the CNN and RNN do not have parameters based on the length of the sequence, but the height must be fixed as the RNN needs a fixed-length vector as input for each time step.

4) Attention Mechanism: For the attention mechanism, we proposed recurrent attention to both characteristics and position. At each step of attention, the mechanism receives the output sequence of the Encoder, with the attention vector at the preceding time step, and at the output of the Decoder at the previous time step as shows Figure 3, and outputs a context vector using the following equations. At each time step T, and at each of the positions L in the sequence, the energy is calculated by EQ. (1):

$$e_{T,L} = w \times tanh(W \times C^{D}_{T-1} + V \times E_L + U \times A_{t-1} + b) \quad (1)$$

where w, b, W, V are trainable network parameters, C^{D}_{T-1} is the output of the Decoder, E_L is the element in the sequence outputted by the Encoder at position L, A_{t-1} is the attention vector at time $t-1$, and b is a bias. A Softmax is calculated

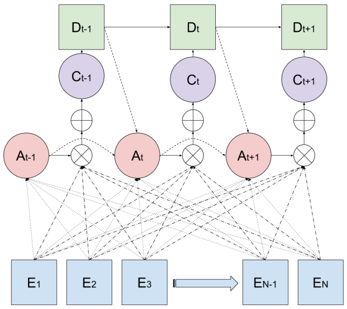


Figure 3. Global computation of the attention mechanism: E_i is the i^{th} element in the sequence outputted by the Encoder, D_j is the j^{th} output of the Decoder.

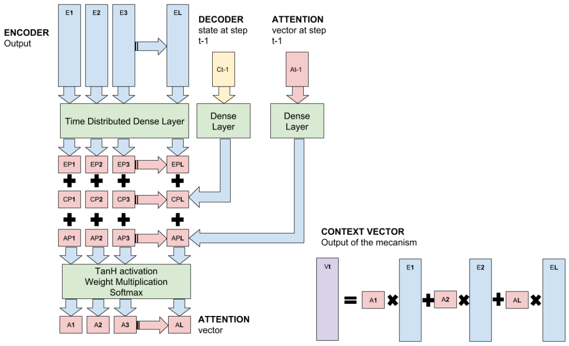


Figure 4. Local computation of the attention mechanism

for this time step T to obtain the standardized attention vector as seen in EQ. (2):

$$A_{t,L} = softmax(e_{t,L}). \tag{2}$$

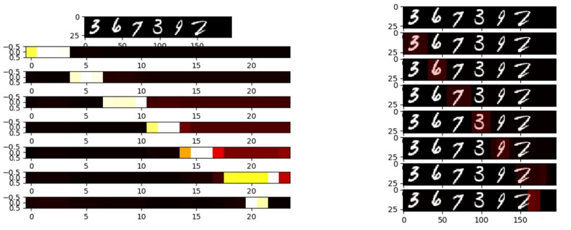
The context vector of this time step T is computed by summing as in EQ. (3):

$$C_T = \sum_i (A_{T,L} \times E_L) \tag{3}$$

These equations are illustrated in Fig. 4. The diagram shows at the top how the attention mechanism takes into account the encoded sequence and the state of the encoder at time $t-1$. This state is used to obtain a value for each element of the sequence that will be added to the values of the sequence. This diagram reproduces the attention mechanism of [2]. Adjusting the Perceptron, responsible for the weight of the mechanism, we achieved very interesting results, as displayed in Figure 5 where we see the attention which follows the figures.

III. Used Datasets

To evaluate the proposed system, we choose the ORAND-CAR A and B and the CVL HDS dataset used in ICFHR 2014 [12].



(a) Value of attention vectors

(b) Vector and image overlay

Figure 5. Attention vector.

Figure 6. Samples from the ORAND CAR databases A and B.

Figure 7. Samples of the CVL HDS dataset.

ORAND-CAR: ORAND-CAR (Courtesy Amount Recognition) is composed of two datasets A and B and contains handwritten digit strings extracted from bank checks. All images have noisy background, and numbers with 2 to 8 digits. Some images have non numeral writing like lines, dots or dashes. The dataset A is composed of 2009 images for the training set and 3784 for the testing set. The dataset B is composed of 3000 images for the training set and 2936 for the testing set.

CVL HDS: The CVL Handwritten Digit String is a dataset composed of 26 different numbers written by 300 different persons. All images have a white background and the color of ink used can vary. The train set is composed of 1262 images with 10 different numbers written, and the test set is composed of 6698 images of 26 different numbers, including the 10 numbers from the training set. Classical methods with explicit segmentation achieved good results on this dataset but all sequence-to-sequence based approaches perform poorly, and one of the reason of this could be the lack of variety in the sequence which prevent the systems to generalize enough. To evaluate our system performance on this dataset we had to:

- Separate the test dataset in two sets: the numbers present in the training set and the rest.
- Shuffle the training and the test set with the same number of images as before, but with all 26 numbers present in the training set.

MNIST: It is the standard for handwritten single digit recognition [13]. For simplicity, we adapted it to our problem by concatenating the digits to create digit strings. As MNIST has no information about the identity of each writer, we could not guarantee an homogeneity in writing style inside the string as it could be observed in the ORAND-

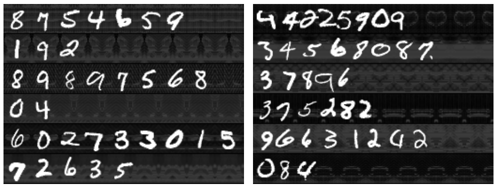


Figure 8. Samples from the MNIST Multi-digits database 1 and 2.

CAR dataset. During the generation, here are all the allowed variations:

- Number of digits composing the string: $[2, 9]$ digits
- Spacing between the digits: $[-14, 0]$ pixels
- Rotation applied to the digit: $[-5\deg, +5\deg]$
- Height re-scaling of all the digit: $[0.7, 1.0]$
- Width re-scaling of all the digit: $[0.5, 1.0]$
- Background: $[Regular, Random, None]$

Height and width re-scaling are the same for each digit in an image but spacing and rotation can vary inside the same image. The regular background is a randomly chosen pattern repeated over all the image. The aleatory background is a Gaussian noise applied to the image before placing the number over it. None is just the same background as MNIST digit: all White or all Black if inverted. Thus, we generated two datasets: MNIST-M1 and MNIST-M2. The difference between M1 and M2 is the number of allowed variations, the generator for M1 is only allowed to vary the number of digits composing each string, with static positioning and no re-scaling or rotation, but the generator for M2 can use all available variations. The training sets are composed of 30000 images generated using only digit image for the MNIST training set, and test sets are composed of 5000 images generated using only digit images for the MNIST testing set.

IV. Experiments and Results

We implemented the attention mechanism as a recurrent cell as it acts in a sequential way and uses the attention vector for the previous time step to compute the attention vector for the current one. As the attention also needs the state of the Decoder which uses the context vector produced by attention, we created one unique recurrent cell combining the attention mechanism and the LSTM cell used as Decoder. We used a computer having a CPU Intel Xeon E5-2620 v4, without GPU and Keras 2.2 with Tensorflow 1.8 back end. Categorical cross entropy was used as loss function during training.

A. *Performance Evaluation*

We used a hard and soft metric to evaluate the performance of our system. The hard metric being the Word Recognition Rate (WRR), validating only if all labels in the strings are truly recognized. The soft metric is the Character Recognition Rate (CRR) which computes the Edition distance (Levenhstein distance) between the string recognized and the labels to get the Character Error Rate (CER) with $CRR = 1 - CER$. It seems that the soft metric is more reliable, to compare the results, than the hard one which is not linear. In fact, an improvement of 5% in the soft metric can result in a improvement of 0 to 20% in the hard metrics depending on where the initial precision stand. But, when we compared with other systems, we only used the hard metric for conformity reasons.

Table II
Character precision rate obtained for each configuration.

CNN layers	Character Precision Rate	Execution time for one Epoch
3	91.53	70s
4	92.13	90s
(5)	92.54	-
6	**92.58**	-
7	91.89	-
MaxPooling width Reduction	-	-
(2)	**95.02**	-
4	92.54	-
8	92.88	-
16	85.64	-
Encoder Layers	-	-
1	91.25	-
(2)	**95.02**	-
3	94.29	-
Decoder Layers	-	-
(1)	**95.02**	-
2	94.87	-

B. *Network Variations*

To find the best architecture for our system, we tried various settings and retained those between parentheses:

- Number of layers inside the CNN: 3, 4, (5), 6, 7
- Reduction of the sequence length: (2), 4, 8, 16
- Number of layers of the Encoder: 1,(2), 3
- Number of layers of the Decoder : (1), 2
- Normalization Function of Attention : (Softmax), Sharpmax, Smoothmax.

During experiments, we used the ORAND-CAR A Dataset with 2000 images per epoch and a batch size of 100. The obtained results are displayed in Table II where retained parameters are between parentheses. We made these variations in the same order as they are presented upside, by testing with all the different values, then fixing these parameters with the best values obtained before switching to the next parameters.

C. *Transfer Learning*

During training, the attention mechanism revealed to be hard to train from scratch and on harder database like ORAND-CAR A or B. The convergence was reached in a sub-optimal way. The attention weights were almost random and the results were not satisfactory. To overcome this

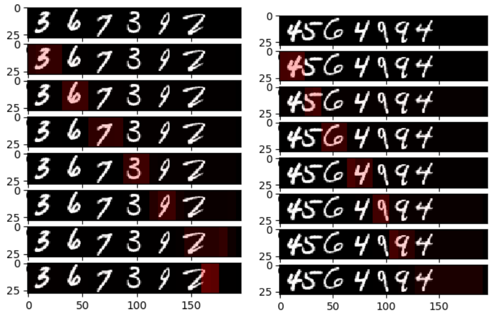


Figure 9. Visualisation of the attention mechanism at each step on one sample from MNIST-Multi1, and one sample from MNIST-Multi2.

Table III
RESULTS ON ORAND-CAR A, ORAND-CAR B AND CVL HDS.

Competitor	ORAND-CAR A	ORAND-CAR B	CVL HDS
Tebessa I*	0.3705	0.2662	0.5930
Tebessa II*	0.3972	0.2772	0.6123
Singapore*	0.5230	0.5930	0.5040
Pernambuco*	0.7830	0.7543	0.5860
BeiJing*	0.8073	0.7013	**0.8529**
Proposed	**0.8401**	**0.82.97**	0.2245

problem, we resorted to the transfer learning to not start from scratch and provide the network with a good starting point. Transfer learning is simply initializing the network with weights obtained by training the network on another dataset. We used the generated datasets MNIST-M1 and MNIST-M2 and achieved better results and a good working attention as intended (see Figure 9). As shown in Table III, the proposed system reaches the highest performances on the public datasets, compared to those of systems in ICFHR 2014 competition (indicated by stars) with a recognition rate 84.01%. Notice that the state of the art mentions two other works using RCNN [16] and ResCNN [6] that exceed this rate but we had no way to verify it.

V. CONCLUSION

In this work, we postulate that handwritten digit string recognition can be successfully done without any segmentation. For that, we proposed a deep network architecture consisting of one CNN and two RNNs, one acting as an Encoder and the other as a Decoder, by resorting to the attention mechanism. Experiments carried on several datasets, whether on numerical chains created artificially from MNIST or directly on datasets proposed in competitions, demonstrate the contribution of the attention mechanism in improving the recognition performance. The different choices of alignment, calculation of energy and contextualization have been beneficial. We show that the proposed system achieve competitive results, compared to those of ICFHR competition. Additional work is necessary to go beyond the scores displayed in the state of the art that we were able to verify in this work. Future work concerns the improvement in the use of attention mechanism and its training to better the alignment. A more dedicated architecture to the digits chains of digits must also be reflected.

REFERENCES

[1] I. Sutskever, O. Vinyals and Quoc V. Le, *Sequence to Sequence Learning with Neural Networks*, Conference on Neural Information Processing Systems, Montreal, 2014.

[2] J. Sueiras and V. Ruiz and A. Sanchez and J. F. Velez, *Offline continous handwriting recognition using sequence to sequence neural networks*, Neurocomputing, 2018.

[3] S. Vogel and H. Ney and C. Tillman, *HMM-based word alignement in statistical translation*, Proc. of the 16th conference on Computational linguistics, 1996.

[4] A. Graves, S. Fernandes, F. Gomez and J. Schmidhuber, *Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Network*, Proc. of the 23rd International Conference on Machine Learning, 2006.

[5] A. Graves, *Offline handwriting recognition with multidimensional recurrent neural networks*, NIPS, 2008.

[6] H. Zhan, Q. Wang and Y. Lu, *Handwritten digit string recognition by combination of residual network and RNN-CTC*, CoRR, 2017.

[7] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, ICPR, 2016.

[8] D. Bahdanau and K.Cho and Y.Bengio, *Neural Machine Translation by jointly learning to align and translate*, ICPR, 2016.

[9] O. Matan, J.C. Burges, Y. LeCun and J.S. Denker, *Multi-digit recognition using a space displacement neural network*, Advances in Neural Information, 1999.

[10] S. Hochreiter and J. Schimdhuber, *Long Short Term Memory*, Neural Computation, Vol. 9, Issue 9, 1997.

[11] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho and Y. Bengio, *Attention-Based Models for Speech Recognition*, Advances in Neural Information Processing Systems, Montreal, 2015.

[12] M. Diem, S. Fiel, F. Kleber and R. Sablatnig, *ICFHR 2014 competition on handwritten digit string recognition in challenging datasets*, ICFHR, 2014.

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

[14] R. Saabni, *Recognizing handwritten single digits and digit strings using deep architecture of neural networks*, ICPR, 2016.

[15] Kim, Suyoun, Hori, Takaaki, Watanabe and Shinji, *Joint CTC-attention based end-to-end speech recognition using multi-task learning*, ICASSP, 2017.

[16] B. Shi, X. Bai and C. Yao, *An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition*, PAMI, 2016.