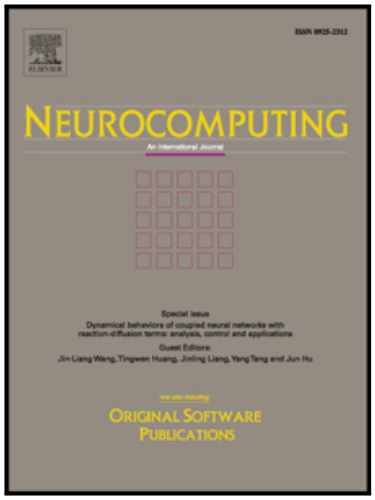
Communicated by Prof. Yicong Zhou

Accepted Manuscript



Offline Continuous Handwriting Recognition Using Sequence to Sequence Neural Networks

Jorge Sueiras, Victoria Ruiz, Angel Sanchez, Jose F. Velez

PII: S0925-2312(18)30137-1
DOI: 10.1016/j.neucom.2018.02.008
Reference: NEUCOM 19298

To appear in: *Neurocomputing*

Received date: 1 August 2017
Revised date: 4 December 2017
Accepted date: 1 February 2018

Please cite this article as: Jorge Sueiras, Victoria Ruiz, Angel Sanchez, Jose F. Velez, Offline Continuous Handwriting Recognition Using Sequence to Sequence Neural Networks, *Neurocomputing* (2018), doi: 10.1016/j.neucom.2018.02.008

Highlights

- A new deep neural architecture to offline handwriting recognition is proposed.
- Convolutional and recurrent layers to get visual and sequence features were used.
- Optimal parametrization of the model was found by systematic experiments
- Best published results are obtained over IAM and RIMES databases with test lexicon.

Offline Continuous Handwriting Recognition Using Sequence to Sequence Neural Networks

Jorge Sueiras, Victoria Ruiz, Angel Sanchez, Jose F. Velez[1]

Rey Juan Carlos University. Madrid. Spain

Abstract

This paper proposes the use of a new neural network architecture that combines a deep convolutional neural network with an encoder-decoder, called sequence to sequence, to solve the problem of recognizing isolated handwritten words. The proposed architecture aims to identify the characters and contextualize them with their neighbors to recognize any given word. Our model proposes a novel way to extract relevant visual features from a word image. It combines the use of a horizontal sliding window, to extract image patches, and the application of the LeNet-5 convolutional architecture to identify the characters. Extracted features are modeled using a sequence-to-sequence architecture to encode the visual characteristics and then to decode the sequence of characters in the handwritten text image. We test the proposed model on two handwritten databases (IAM and RIMES) under several experiments to determine the optimal parametrization of the model. Competitive results above those presented in the current state-of-the-art, on handwriting models, are achieved. Without using any language model and with closed dictionary, we obtain a word error rate in the test set of 12.7% in IAM and 6.6% in RIMES.

Keywords: Artificial Intelligence, Handwriting Recognition, Convolutional Neural Networks, Recurrent Neural Networks, Sequence to Sequence Networks

[1]Corresponding author. E-mail address: jose.velez@urjc.es (J. Velez).

1. Introduction

The offline continuous handwritten text recognition problem [1] consists in the automatic transcription of images that contain handwritten text. The problem has been extensively studied in the literature [2] [1] [3] [4]. Handwriting recognition presents relevant applications such as bank check processing, automatic address reading [5] or content extraction from digitalized databases of historical documents [6]. Despite of recent advances, significant differences in the writing of individuals and imprecise nature of handwritten characters make this recognition task hard and it still remains as an open research problem.

The problem of handwriting recognition is very complex and usually includes several steps [3]: text line detection and segmentation, breaking text line into words and word recognition. In this paper we put the focus into the word recognition problem. This is, we present a model that recognize the handwriting words that appear in an image.

Before the irruption of deep neural networks in 2009, the best results for recognition of continuous offline handwritten texts were obtained by using Hidden Markov Models (HMM) techniques [7]. Since 2009, recurrent neural networks and deep feedforward neural networks (in particular, the bi-directional and multi-dimensional Long Short-Term Memory or LSTM) have won several international handwriting competitions [1][8].

Deep Convolutional Neural Networks (CNN) provide current state of the art results in handwriting isolated character recognition [9]. In this paper, we propose to take advantage of using a CNN to extract relevant features from word images. We aim to extract relevant features in order to identify the characters included in the word, without needing to segment it into characters.

Next, we use these CNN extracted features as the input to a Sequence to Sequence model (Seq2Seq). Seq2Seq architectures are designed to solve problems that need transform one input sequence signal into other output sequence signal of different length. These architectures have been proven to successfully solve complex problems, such us machine language translation [10], speech recogni-

tion [11] and lip-reading [12]. Based on these architectures, we propose a new Seq2Seq model for the offline handwriting problem.

The main contribution of this paper is the novel architecture proposal for the offline handwriting recognition problem. This architecture combines a CNN with a Seq2Seq model. The CNN is applied over a sequence of image patches obtained from the word image using a horizontal sliding window. So, the CNN models the visual attributes of the handwritten words and provides a sequence of visual features from each part of the word image. This feature sequence is used as input to the Seq2Seq model. Across these inputs, the Seq2Seq network identifies the characters of the word using its encoder-decoder functionality.

In addition, we analyze different parametrizations in the proposed architecture and identify the most relevant ones to optimize the accuracy of the model. Finally, the generalization capacity is validated using two different corpora in different languages, and our results are compared to other related approaches.

This paper is focused on Latin character languages, although we think that the proposed solution could be used on other character sets. Our approach has been applied on the character dictionaries available for the two corpora analyzed (IAM [13] and RIMES [14] databases). These databases include English and French languages with uppercase/lowercase letters, digits and some punctuation signs, but they do not cover the complete ASCII printable character table.

2. Related work

Over the last decades, different approaches have been applied to the handwriting word recognition problem (see Plamondon et al. [3]). The main differences between the authors appear in two key elements of the proposed solutions: a) the strategy to extract features from the handwriting image, and b) the way to decode the output of the classifier to predict the sequence of characters which form part of a given word.

Two main strategies are commonly used to extract image features: applying different Computer Vision techniques to detect them or use directly the pixels

columns of the image as raw features. The first strategy is applied by Doetsch et al. [15] and Kozielski et al. [16] who employ Principal Component Analysis (PCA) to extract components over fixed-size frames of 8×32 pixels. Bideault et al. [17] also extract features, in this case using Histograms of Oriented Gradients (HOG). Graves et al. [1] consider characteristics extracted of each column pixel, such as mean and other moments, center of gravity, transitions and other aggregations. In the other hand, the works by Graves et al. [1] and Bluche et al. [18] use directly the pixel features as input of the handwriting model.

There are also two main options to decode the handwriting prediction output for converting it in the sequence of characters that identifies the handwritten words. The first one consists in using an HMM (see for example Bluche et al. [19] and Doetsch et al. [15]), which is the most traditional way to decode the word. The second way is using the Connectionist Temporal Classification (CTC) objective function introduced by Graves et al. [20]. Several authors also apply this second option (see Graves et al. [1], Bluche et al. [18] and Voigtlaender et al. [21]).

In the last years, the main model architectures applied to the offline handwriting problem are Recurrent Neural Networks (RNN), specially Long Short-Term Memory (LSTM) networks. Several authors have proposed RNN-based models with different strategies to extract features and different decoding processes. Kozielski et al. [16] applied PCA features over an architecture that included a HMM baseline layer followed by a LSTM layer with a final HMM decoder. Some other works, like [1], [21] and [18], proposed a RNN variant named Multidimensional Recurrent Neural Network (MDRNN), introduced in Graves et al. [22], which exploits the bidimensional nature of the handwritten images. In all cases, the authors proposed a deep architecture with several Multidimensional Long Short Term Memory (MDLSTM) layers and the CTC decoder at the top. Doetsch et al. [15] proposed a LSTM modification of the activation functions inside the gated units over PCA extracted features and using a HMM decoder. Stuner et al. [23] used two architectures: a Bidirectional LSTM with two layers and a MDLSTM architecture with three layers. Finally, Sabatelli and

Shkarupa [24] have proposed in 2016 a Seq2Seq model to recognize Gothic handwritten texts with two main differences over our approach: they neither used convolutions to extract features from the handwriting images, nor an attention mechanism into the Seq2Seq architecture.

3. High level description of the word prediction model

In this section we describe the model architecture at high level. It is based on the Seq2Seq approach, introduced by Sutskever et al. [10]. The model transforms a variable-length sequence of handwritten word image slides into the variable-length sequence of the characters that conform the previous word.

The proposed architecture is inspired by two of the main cognitive psychology models relative to handwriting word recognition: the Serial Letter Recognition (SLR) and the Parallel Letter Recognition (PLR) models [25]. The SLR model states that words are read letter by letter, serially from left to right. On the other hand, the PLR model states that letters within a word are recognized simultaneously, and this joint information is used to recognize the word. The advantage of our architecture is that it considers the two previous models, and so, it decodes a word character by character but considering the whole input into each decoded character.

Our proposed model has three main components: a convolutional reader, an encoder and a decoder with an attention mechanism. These components are combined into a Seq2Seq architecture as shown in Fig. 1.

The convolutional reader function C is a deep CNN that converts each input image patch p into a vector of visual features x. The inputs to this convolutional reader consist in a sequence of patches obtained from a word image. Note that some patches can include a part of a letter or an intersection between two letters. The convolutional model extracts feature vectors of features of these patches related to the letters that appears in it. So, the convolutional reader is applied over a sequence of patches generating a sequence of convolutional features.

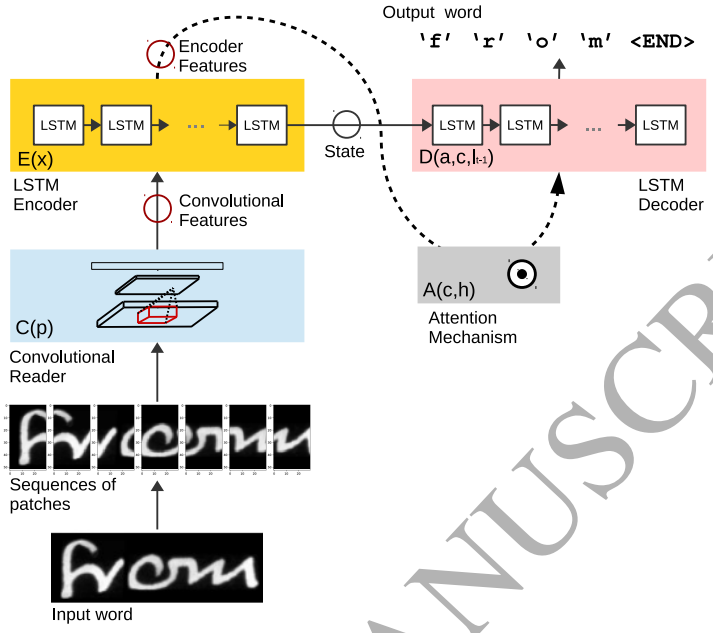


Figure 1: Components of the high-level model architecture for word recognition

Next, the encoder E function, that is a LSTM, reads the sequence of convolutional features x and extracts the relationships among these features. So, the encoder transforms x into a sequence of encoded output features h that model the relationships among the letters of a word. In addition, the final state c of the encoder LSTM was preserved to use as initial state in the LSTM decoder.

The attention mechanism A helps to put the focus on specific parts of the encoder output features h that are relevant for specific output letters. For example, the first letters need to put the attention in the first elements of the encoder output features, that represent the first part of the word image. On the other side, the last letters need to put the attention in the final part of the encoder features, that are associated with the last part of the word image.

Finally, a decoder D gets the previous encoder output transformed by the attention mechanism A to predict letter by letter the word. The decoder consists in a LSTM that generates each current letter l_t combining the state vector c, the encoder outputs transformed by the attention mechanism a and the previous

letter l_{t-1}.

We can summarize all the previous processes by the following sequence of equations:

$$\begin{aligned} x &= C(p) \\ (c, h) &= E(x) \\ a &= A(c, h) \\ l_t &= D(a, c, l_{t-1}) \end{aligned} \tag{1}$$

4. Architecture and algorithmic details

In this section, we describe the implementation of the proposed model and the relevant details about its components. We start with a description of the data preprocesing, which facilitate the recognition task of handwriting text. Next, we continue with a detailed description of the Seq2Seq architecture used by our model. In this description we explain the details of the five model components: convolutional reader, LSTM layers, encoder, decoder and attention mechanism, respectively. Fig. 2 shows a detailed schema of the complete model architecture. This figure follows the notation introduced in Section 3.

Finally, we point out some relevant implementation details including: hardware, libraries, parametrization of the optimization algorithm and data augmentation strategies.

4.1. Data Preprocesing

The data used in our experiments were preprocessed to correct some usual characteristics of the handwriting text that difficult the recognition task. This preprocessing was applied at line-of-text level. In particular, we identified the baseline and the corpus line, corrected the line skew, corrected the slant and normalize the height of the characters based on baseline and corpus line.

To detect the baseline we applied a Random Sample Consensus (RANSAC) regression, proposed by Fischler et al. [26], over the lower points of each column in the image of a given line. We applied the same method over the higher points

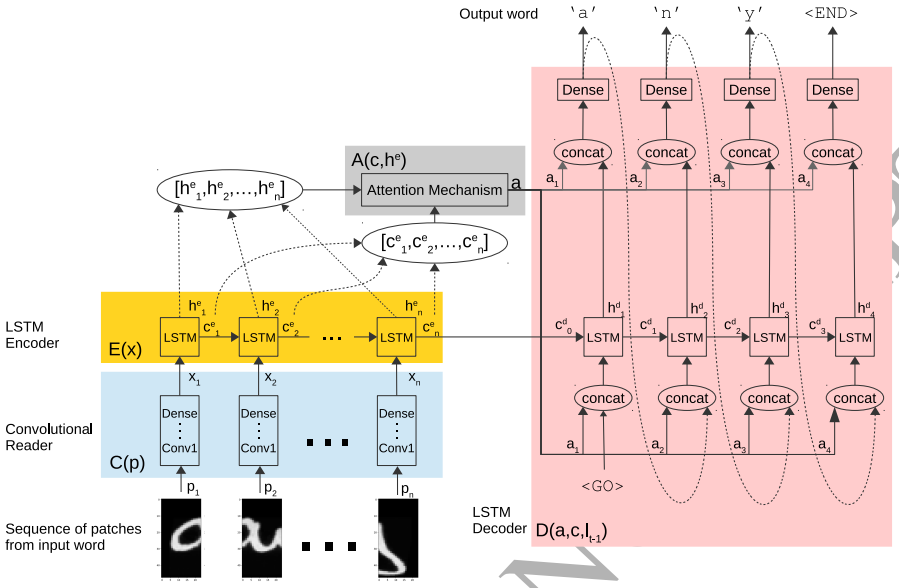


Figure 2: Detailed model architecture

of each column to identify the corpus line. To correct the skew, we computed the baseline angle to apply an affine transformation over the line image. To correct the slant, we first identified the slant angle α using the algorithm detailed in [27]. Next, by knowing the slant angle α and the image width w, we corrected the slant using the affine transformation matrix defined by Eq. (2).

$$\begin{pmatrix} 1 & -\alpha & 0.5 \cdot w \cdot \alpha \\ 0 & 1 & 0 \end{pmatrix} \tag{2}$$

Finally, we adjusted the height of the characters by resizing it over the corpus line and under the baseline to a half of the height of the main part of the line (the part which is located between the baseline and the corpus line). Fig. 3 shows one example of an original line of the IAM database and the result produced after the described normalization process.

Next, we extract, for each original word image, a sequence of horizontal overlapped slides. These slides can include a part of a letter, one letter or more than one letters. These slides are the inputs to our model, and they enable the

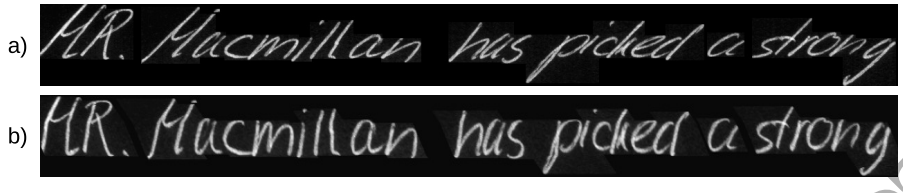


Figure 3: Line preprocessing example: (a) Original line and (b) preprocessing results.

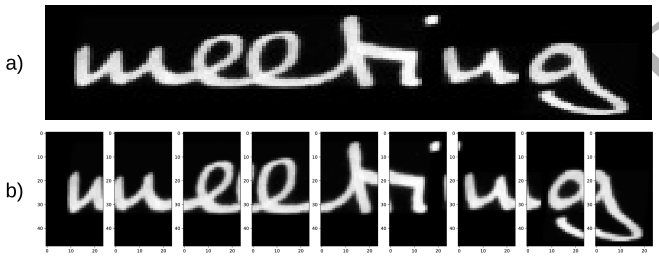


Figure 4: Word patch extraction: (a) Original image word and (b) extracted input model patches.

model to view the word image not globally but as a sequence of patches. These patches are characterized by two parameters: the patch size (i.e. its horizontal size) and the patch step (i.e. the number of pixels that we move to the right to extract the next patch). Fig. 4 shows an example of an original word image and its corresponding extracted patches.

4.2. Model components

The proposed solution is designed to model each letter l_i in the word $w = (l_1, l_2, ..., l_n)$ as a conditional probability of the input image patches $Patches = (p_1, p_2, ..., p_m)$ and the probability distribution of the previous letters of the word l_j with $j < i$.

$$\mathbf{P}\left(l_i | Patches, l_{<i}\right) \tag{3}$$

Therefore we model the output probability distribution of a word given the sequence of patches, as follows:

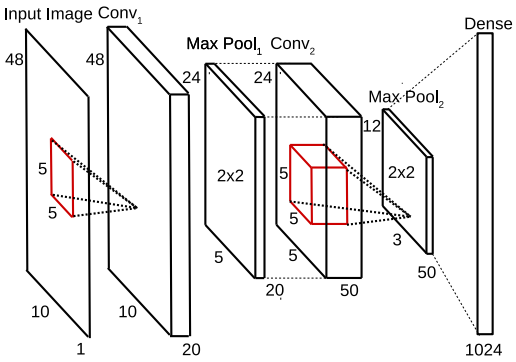


Figure 5: CNN architecture schema used in the convolutional encoder component

$$\mathbf{P}\left(w|Patches\right) = \prod_{i} \mathbf{P}\left(l_i|Patches, l_{<i}\right) \tag{4}$$

In the next subsections, we detail all the elements of the architecture that implements this model.

4.2.1. Convolutional reader

The objective of the convolutional reader is to extract the visual features which are relevant to identify the characters appearing in the input patches of the word image.

Our model uses a deep convolutional architecture (CNN) stacking two groups of one convolutional and one max pooling layers and finishing with one dense layer. Figure 5 shows the detailed CNN architecture of the model proposed. This architecture was inspired in the LeNet-5 architecture, published originally by LeCun et al. [28] to solve the problem of identifying isolated handwriting characters. Similar architectures achieved good results to solve the handwriting character identification problem over different handwritten character databases [29].

We apply this architecture model to each patch of the input sequences extracted from a word image. From each patch, the CNN computes a vector that contains the visual features related to the character or characters included in

Table 1: Comparative of WER results for different dense layer sizes of the convolutional reader.

Database	Dense size		
	512	1024	2048
IAM	21,1	**18,9**	Not convergence
RIMES	13,6	**12,8**	13,2

it. The image patches are the inputs, and the convolutional features obtained from the last dense layer are the outputs. We tested three different sizes for this final dense layer (respectively, with 512, 1024 and 2048 neurons). These tests were made over the optimal architecture obtained in section 5. They let us to identify that the 1024 size shows the best WER over the validation set. We detail the results of these experiments, calculating the WER on the standard lexicon introduced in subsection 5.3, in Table 1.

Following the notation of Fig. 2, the output of this CNN is a sequence of vector features $x = (x_1, x_2, ..., x_n)$ that encapsulates the visual features of the patches extracted from a handwritten word image.

$$x_i = C(p_i) \tag{5}$$

4.2.2. LSTM layers

Long Short-Term Memory (LSTM) networks are a special type of Recurent Neural Networks (RNN) which are able to learn long-term dependencies. Fig. 6, provided by Graves et al. [30], includes a schema of the LSTM cell. LSTM networks were introduced by Hochreiter and Schmidhuber [31] and include two ways to translate the previous information across the net: the output vector h and the state vector c, that combined using three gates, are explicitly designed to store and propagate long-term dependencies.

The gate i, which is named the input gate, learns which values will be updated in the state vector. The gate f, which is named the forget gate, learns the information from the previous state that can be thrown away. With the

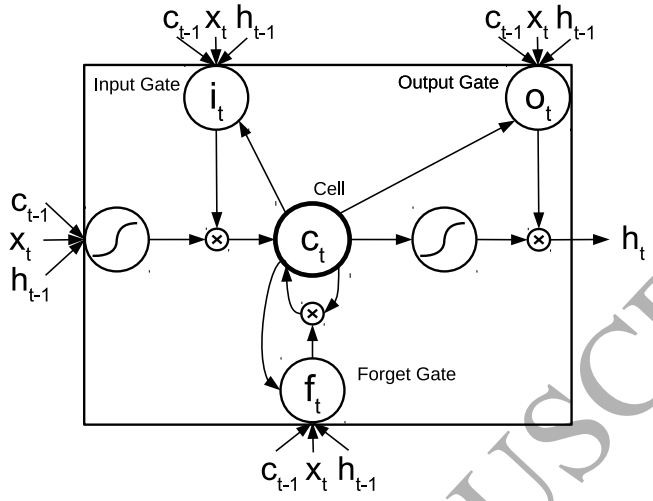


Figure 6: Long short-term memory cell

output of these two layers the network creates a vector of new candidate state values c. Finally, the gate o, which is named the output gate, learns what information will go to the output h. The following equations model in detail the LSTM:

$$\begin{aligned} f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \\ i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \\ c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \\ o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \\ h_t &= o_t \tanh\left(c_t\right) \end{aligned} \tag{6}$$

where W and b are trainable parameters, σ is the sigmoid function, x is the input sequence, i is the input gate, f is the forget gate, o is the output gate, c is the state vector and h is the output vector.

4.2.3. Encoder

The goal of this component is to capture the horizontal relationships existing among the input data, which consist in a sequence of overlapped horizontal

patches of handwritten word images. To do that, the encoder receives the sequence of vector features x_t, that represent each input image patch p_t generated by the convolutional reader, and generates two outputs: a sequence of state vectors c_t and a sequence of output vectors h_t, each one for each encoder step t.

$$< c_i^e, h_t^e >= LSTM(x_t^e, c_{t-1}^e, h_{t-1}^e) \tag{7}$$

The encoder is implemented using LSTM layers. We experimented with different LSTM architectures for the encoder. We try it using one LSTM layer, two LSTM layers stacked and even changing simple LSTM layers by bidirectional LSTM layers using one, two and three stacked layers. We also experimented with the size of the layers in order to identify the optimal architecture for the encoder. In subsection 5.4 we detail the results of these experiments.

4.2.4. Decoder

The decoder is also based on a LSTM network combined with an attention mechanism. For each decoding step t the LSTM generates a state vector c_t^d and an output vector h_t^d using as input the context vector a_t provided by the attention mechanism, concatenated with the probability distribution of the previous letter predicted l_{t-1}. The initial state of the decoder c_0^d is initialized with the last state of the encoder c_n^e and the initial letter is initialized by the special character $< GO >$.

$$\begin{aligned} &\alpha_t = A(h_t^e, c_t^e) \\ &< c_t^d, h_t^d >= LSTM(concat(l_{t-1}, a_i), c_{t-1}^d, h_{t-1}^d) \\ &c_0^d = c_{last}^e \\ &l_0 =< GO > \end{aligned} \tag{8}$$

The probability distribution of the output character for each step is generated by a *dense* layer with the $softmax$ activation function applied over the output sequence concatenated with the attention mechanism output.

$$P(y_i|X) = softmax(dense(concat(h_i, a_i))) \tag{9}$$

The decoder starts with the special signal $< GO >$ as input to decode the first character of the word image. It ends when at the output appears the special character $< END >$, by selecting the sequence of previous output characters as the decoded word.

The recurrent nature of a LSTM decoder will try to predict the next character of a word using the prediction of the previous character. This means that the decoder could be acting as a character level language model. However, our model is trained at word level and this fact limit considerably the capacity of the decoder to act as a true language model (a true character language model typically needs characters of the previous words to capture the grammar). For this reason, we made that the decoder prediction depends also on the final encoder state and on the output of an attention mechanism, which will be described in the next subsection. These two components will provide to the decoder the additional features needed to predict correctly the output characters.

4.2.5. Attention mechanism details

Our model uses the attention mechanism proposed by Chorowski et al. [11]. This mechanism allows the decoder to access to the features of the LSTM encoder at every decoding step. For each step, we calculate the attention states as a function of the outputs and the states of the encoder. This allows that each step of the decoder puts its attention in a specific part of the output of the encoder. This is very important on the architecture, because it allows the decoder to play attention to the input data part that is more relevant to decode the current letter. For example, if the decoder is estimating the first letter of the word, it can put the attention in the features that are generated by the first part of the word image. And so on, with all the letters to be predicted.

The detailed equations of the attention mechanism are:

$$\begin{aligned} e_{t,l} &= w^T \tanh(W c^e_{t-1} + V h^e_l + b) \\ a_{t,l} &= \frac{\exp(e_{t,l})}{\sum_{l=1}^{L} \exp(e_{t,l})} \end{aligned} \tag{10}$$

where w and b are trainable weight and bias vectors, W and V are trainable

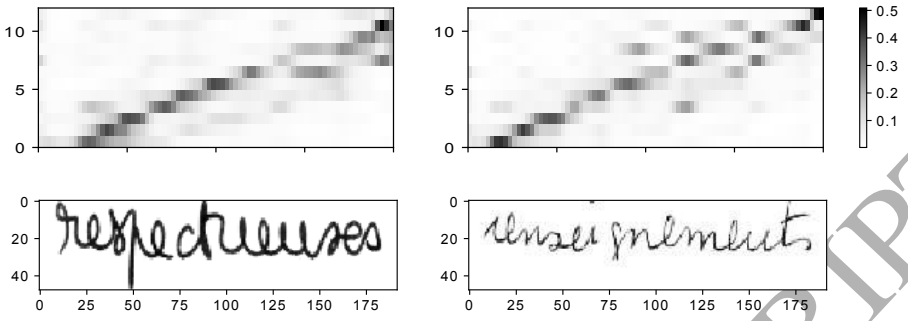


Figure 7: Example of attention mechanism outputs for two words and their respective original handwritten word images.

matrices, c^e is the encoder state and h^e is the encoder output. Finally, l runs from the first output letter to the last letter L. The attention mechanism is implemented as a neural network layer that combines the states and the outputs of the encoder with the hyperbolic tangent as activation function. Then, we apply the $softmax$ function to normalize the columns of the matrix E, in order to add one by column in the attention outputs $a_{t,l}$.

Reading a handwritten word could be considered as a monotonous task, because you need to read the left letters before the right ones. But this fact is not true for algorithms. Reading letters of a word, written using a latin script, from left to right is a conventionalism, and it is possible reading them from right to left without loss of information. So, to allow the model considering all the neighborhood information, the selected attention mechanism has no monotonicity restrictions.

To illustrate this assumption, we plot in Fig. 7 the outputs $a_{t,l}$ of the attention mechanism for two words of the validation partition. These figures present in the horizontal axis the image patch number and in the vertical axis the predicted character position. The figures show that, in general, the attention mechanism is working monotonically, but for some character positions it uses information from patches at the left of what will be a monotonous behavior.

5. Evaluation results

In this section, we describe the evaluation results of the model applied on two well known handwritting datasets. The IAM offline handwritting [13] and the RIMES databases [14].

Two metrics were used to evaluate the model: the Character Error Rate (CER), measured as the Levenstein distance between the predicted and the real character sequence of the word, and the Word Error Rate (WER), defined as the percentage of words incorrectly identified. We evaluated these metrics with the direct output of the model and also after the application of a basic language model based on dictionary search in standard lexicons.

In the process of optimizing the selected architecture, we performed several experiments by changing the parametrization of the model to determine the effects in the error of changing different parameters. In subsections 5.4 and 5.5 we describe the results of these experiments.

We compare the best results obtained by the proposed model with other approaches over the same databases and with the same lexicons, when it is possible. These results are included in subsection 5.6

Finally, we dedicate a final subsection to analyze the nature of the error by investigating the error types that we got, in order to propose possible improvements to the current model.

5.1. Databases overview

The IAM database was introduced by Marti et al. [13]. It consists on several form images of handwritten English text. The database is divided in train, validation and test partitions and it has been used extensively in the literature. See for example Doetsch et al. [15], Bluche et al. [18] or Graves et al. [1]. The partition consists on 6,482, 976 and 2,915 lines for the training, validation and test sets, respectively. That means a total of 55,081, 8,895 and 25,920 words in each partition. For the analysis, we selected the words marked that are segmented correctly in the IAM database, with this selection we obtain

a final 47,952, 7,558 and 20,306 words for the training, validation and test sets, respectively

The RIMES database [14] consists on images of handwritten letters from simulated French mail. We used the ICDAR 2011 competition partition provided by the A2IA Lab, which is the maintainer of the database. It includes 51,739 words to train, 7,464 words to validation and 7,776 words to test, respectively.

5.2. Implementation details

To use our framework with these two datasets (IAM and RIMES), we need to re-size the word images to a fixed height of 48 pixels and a fixed width of 192 pixels. As we want to keep the aspect ratio of the words, usually we need to complete the images with white pixels in the right part of them.

First, we extract the sequence of overlapped image patches. Due to the height of the images, all the patches have a height of 48 pixels. We experimented with different widths for the patches and with different levels of patch overlapping. The results of these experiments are presented in subsection 5.4

In addition, we also performed experiments with the layers and size of the encoder using unidirectional and bidirectional LSTM, one or two layers and different layer sizes (from 64 to 512). The results of these experiments are presented in subsection 5.5. In all the experiments the model weights are randomly initialized. For the decoder, we used a single unidirectional LSTM of the same size of the encoder LSTM layers.

To implement our solution we used the TensorFlow library [32] and a GeForce Titan X Pascal GPU. As trainer we used the Adam stochastic optimization algorithm, with a batch size of 256 and with a dropout of 0.5 [33] in the dense layers of the convolutional part and in the cells of the encoder LSTM layers. An initial learning rate of 0.001 was used and this rate was decreased by 2% in each epoch. We applied a stopping criteria of 10 epochs without any improvement in the Word Error Rate (WER) on the validation set.

In each epoch, we randomized the order of the training data. In addition, we implemented an online randomized data augmentation process that included:

±10 percent of zoom, ±10 percent of displacement, ±10 degrees of skew, and 3×3 dilation and erosion transformations.

The different experiments took between 60 and 120 epochs. Depending on the stopping criteria and the model parameters, each experiment lasted between 10 and 20 hours.

5.3. Lexicons used

We provide different results for each considered database (IAM and RIMES). In particular, we offer three main results for each experiment. First, the results obtained by the direct application of the model (without any lexicon usage); second, the results using the standard validation lexicons mainly used in the literature; and third the results using the test set lexicon.

In the case of the IAM database, the standard validation lexicon used was the 50,000 most frequent words of the Lancaster-Oslo/Bergen (LOB) [34] and the Brown [35] corpora, when previously the IAM paragraphs from the Brown corpus were removed. In the case of the RIMES database, we use as standard lexicon the set of all the words present in the train and validation partitions.

With the previous settings, three different groups of results are shown: the raw accuracy of the visual model without any lexicon, the improvement obtained when using search over the standard lexicon of each database, and finally, the error obtained when eliminating the out-of-vocabulary (OOV) words in the test lexicon.

5.4. Analysis of the image segmentation strategy

In this first set of experiments, we analyzed the effect over the final model accuracy of the different parametrizations corresponding to the initial process of image patches creation described in subsection 4.1. We considered two main parameters: the patch width and the step size to create the patches. For example, a patch width of 5 and a step size of 2 means that the first patch is created with the columns from 1 to 5, the second patch with the columns from 3 to 7, and so on.

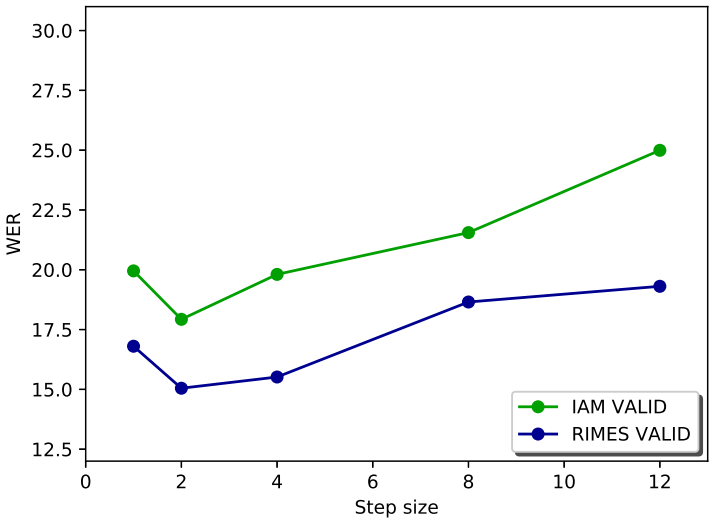


Figure 8: WER in validation and test sets vs step size for IAM and RIMES databases. Minimal WER was achieved with a step size of 2.

We started analyzing the step size with a fixed width of 10 and a fixed encoder architecture of a bidirectional LSTM with 2 layers and a layer size of 256. Fig. 8 shows that for both databases a larger step size produces a larger error, and that the better validation error is achieved with a step size of 2.

So, we determined an optimal value of 2 for the step size, and a fixed encoder architecture of a bidirectional LSTM with 2 layers of size 256. Because we used a convolutional architecture over the patches, the patch width must have a minimal size to apply the convolutions. This minimal width is 5, because the initial convolution layer has a filter of size 5×5. We tested sizes between 5 and 25 in intervals of 5. Fig. 9 shows that smaller or larger patch sizes produced a greater error, and the minimum error was achieved for the patch size of 10 pixels.

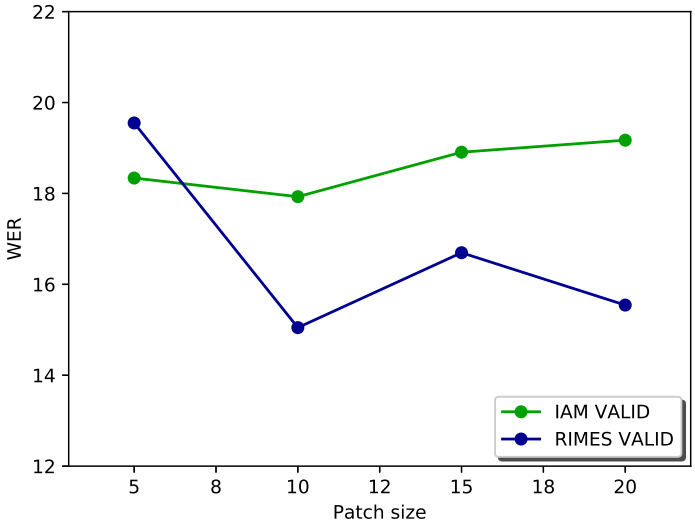


Figure 9: WER in validation and test sets vs patch size for IAM and RIMES databases. Minimal WER was achieved with a patch size of 10.

5.5. Analysis of the LSTM encoder parametrization

Next, we analyzed the effect of the encoder LSTM architecture and its size on the final error. To do this, we considered four different architectures for the encoder component that included unidirectional LSTM with 1 and 2 layers and bidirectional LSTM with 1, 2 and 3 layers. We also experimented with different sizes for the LSTM layers. In particular, we experimented with 64, 128, 256 and 512 cell sizes.

In the first experiment we used a fixed encoder size of 256 and the optimal step size and patch width found in the previous experiments. Fig. 10 shows that best error results were obtained with an encoder architecture of two bidirectional LSTM (BLSTM) layers [8].

For the previous optimal architectures, we experimented with different LSTM cell sizes for the encoder and obtained the results shown in Fig. 11. We observed that optimal result is achieved with the 256 cell size.

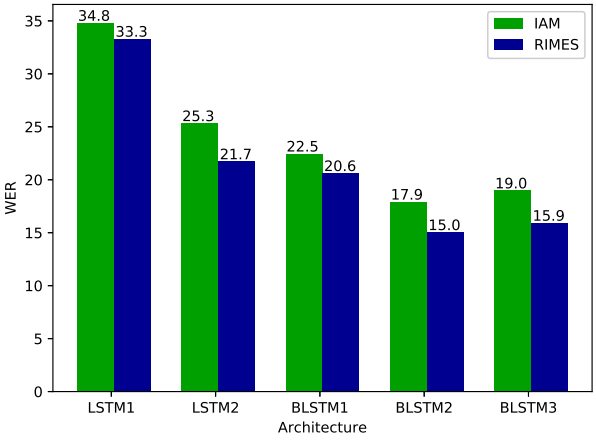


Figure 10: WER in validation and test sets vs encoder architecture for IAM and RIMES databases. Minimal WER achieved in 2 layers BLSTM architecture

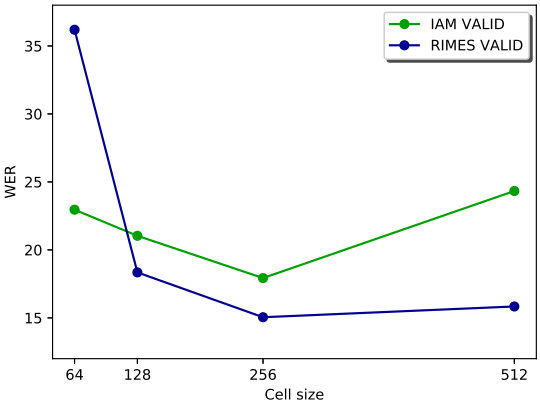


Figure 11: WER in validation and test sets vs LSTM encoder cell size for IAM and RIMES databases. Minimal WER was achieved with a cell size of 256.

In summary, the optimal parametrization for the both IAM and RIMES databases is a model that uses patches with 10 pixels of width and a step size of 2 pixels, and with an encoder with a cell size of 256 and two bidirectional LSTM layers.

5.6. Comparative with other approaches

This work proposes a novel visual model to solve the problem of handwriting word recognition. The fair comparative with other visual models proposed by other authors must be in equal conditions of image pre-processing and prediction post-processing. But this is very difficult because different authors use their own processes, that include specific image pre-processing and specific language models for the post-processing, and usually only provide their final results .

The pre-processing stage is the minor problem, because practically all the authors use the same databases, with the same partition in train, validation and test. So, they perform a similar word image correction, that includes some type of contrast enhancement, and skew and slant corrections. That is the reason why we expect similar results in the final corrected word images. However, in the post-processing we find more difficulties, because the usage of different language models (i.e. at word or at character levels) which combined with different lexicons, can produce very different error results.

In order to provide a fair comparative with other authors, centered in the performance of the visual models, and to facilitate the comparative to future authors, we present three sets of error results. First, the direct error of the visual model without post-processing. Second, the error using direct search in the standard lexicons described in subsection 5.3. Third, the error using the test set lexicon in order to identify the error component produced by the out-of-lexicon words. For all of them, we provide the WER and CER errors over the test set, and a confidence error interval, calculated with 10 repetitions of the optimal model.

For each of the three sets of errors results, we find in the literature other results that use the same databases, the same partitions, a similar word image

Table 2: Comparative of results of the visual model without lexicon.

Database	IAM		RIMES	
	WER	CER	WER	CER
Our method	**23.8±0.05**	8.8±0.02	15.9±0.4	4.8±0.1
Pham et al. [33]	35.1	10.8	28.5	6.8
Bluche [37] MLP	54.2	15.6	59.5	17.8
Bluche [37] RNN	24.7	**7.3**	20.9	5.6
Bluche [36]	24.6	7.9	**12.6**	**2.9**

pre-processing corrections, and one of the three previous post-processing lexicons. We show the comparatives in Tables 2, 3 and 4, respectively, in terms of WER and CER in the test set. The differences observed are caused mainly to the different visual models used.

First, we compared the WER and CER results obtained directly by our visual model with optimal parametrization for the validation set. In all cases, the selected parametrization for our model was the previously identified as optimal in the validation set in the experiments described in subsections 5.4 and 5.5 respectively. In summary, a step size of 2 pixels, a patch width of 10 and an encoder architecture of 2 bidirectional LSTM of size 256 were used. The results are shown in Table 2. The visual models by Pham et al. [33] and by Bluche et al. [36] were a multidimensional LSTM architecture with a CTC decoder extensively used in handwriting recognition with excellent results. Bluche [37] proposed two visual models on pixels features: first a deep MLP/HMM architecture and next a RNN/HMM architecture. Our proposed model outperforms significantly the MLP/HMM and the RNN/HMM architectures, and provides competitive results respect to the MDLSTM/CTC architecture obtaining the best result for the WER in IAM database.

In the second comparative, we process the outputs of the visual model finding the closest word in terms of Levenstein distance over standard lexicons for IAM and RIMES databases, extensively used in the handwriting literature and

Table 3: Comparative of results of the visual model with a standard lexicon.

Database	IAM		RIMES	
	WER	CER	WER	CER
Our method	19.7±0.03	9.5±0.03	**13.1±0.2**	5.7±0.1
Bluche [37] MLP	25.5	8.0	26.1	7.2
Bluche [37] RNN	**16.7**	**5.3**	16.4	**4.3**

Table 4: Comparative of results of the visual model with the test lexicon.

Database	IAM		RIMES	
	WER	CER	WER	CER
Our method	12.7±0.03	6.2±0.02	6.6±0.2	2.6±0.1
Almazan et al. [39]	20.1	11.2		
Bluche et al. [40]	20.5		9.2	

described in subsection 5.3. The results can be viewed in Table 3. In this case the available results are provided by Bluche [37] with the previous deep MLP/HMM and the RNN/HMM architectures. In this case, results are also competitive obtaining the best WER result on RIMES database.

In the third comparative, we used the test lexicon as a dictionary to find the closest word. Although this measure can not be applied to evaluate the unconstrained performance of the system , with this metric we can evaluate the impact in the error of the out-of-vocabulary (OOV) words not present in the standard lexicons. The results are presented in Table 4. In this case, we obtained a significant reduction of WER and CER errors, with competitive results for both databases. Poznanski et al. [38] also used the test set lexicon, obtaining better results that those shown in Table 4. However, these authors used a limited set of characters in their results for the two databases. This makes their results not comparable with those shown in Table 4.

The big differences between the errors with standard lexicons and with test lexicons point out that the OOV errors are very important. The respective

difference of 7.0 points in the IAM error and 6.8 points in the RIMES error can be due to OOV errors.

Only few authors provide results with the test lexicon, this makes it difficult to get a good comparative in this part. To complete it, we calculate for the RIMES database the results obtained by our optimal model applied over the the three proposed subtasks on the ICDAR 2009 handwriting word recognition challenge [41]. The three subtasks corresponding to three sizes of dictionaries: a 99 words dictionary chosen randomly among test words, the test words dictionary and the train + test words dictionary. For this three subtasks our optimal model obtained a WER of 98.30%, 93.77% and 92, 17% respectively.

5.7. Error analysis

This section analyses in detail the errors of the presented models in order to identify possible improvement lines of work. The first analysis was performed in the previous subsection, by studying the influence of the OOV words in the error. These errors can be reduced by improving the lexicons and, specially, by adding to the system some language models at character level that can identify OOV words.

First, we analyze how the errors change in terms of the word length. Fig. 12 shows the WER with the standard lexicons in the IAM and RIMES databases respect to the word length. We observe that the error increases with the word length. This is related with the previous analysis on the OOV influence in the errors, because the words with less letters are articles, prepositions and similar words that typically appears in the lexicons.

To eliminate the influence of the OOV words in the analysis of the error versus the word length, we show in Fig. 13 the WER test error using the test lexicon. We observed that effectively most of the errors in long words are originated by the absence of the correct word in the lexicon. Now, the highest percent of errors appears for word length of 4 in the IAM database and for word length of 5 in the RIMES database.

Next, we analyze the amount of errors originated by the replacement of only

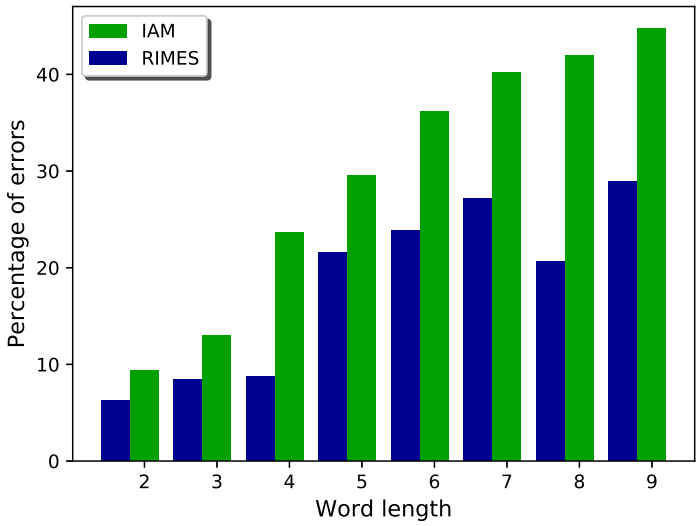


Figure 12: WER in test vs word length.

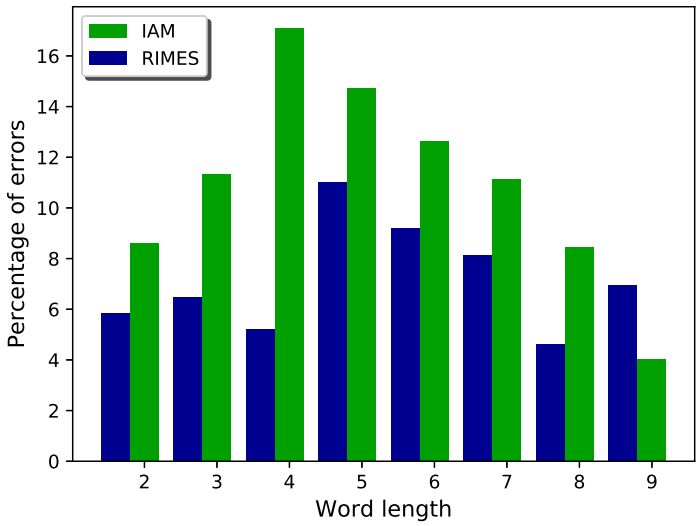


Figure 13: WER in test vs word length with the test lexicon.

one character in a word. That is, the real and predicted words have the same characters and the same length, but in the predicted word it has been replaced one character by other. In this case, we observed that the 68.6% of errors of the visual model in the IAM database and the 69.3% in the RIMES database are errors corresponding to only one character replaced. Probably, these errors could be solved in most cases using a character-level language model.

Finally, we analyze the errors originated by the confusion between the same character but one in lowercase and the other in uppercase. In this situation, we observe that a 5% in the IAM database and a 4.7% in the RIMES database are errors originated by the confusion of some character in uppercase and the same character in lowercase or vice versa. As we expected, the characters where it appears this confusion are those one with a similar appearance in uppercase and lowercase. For example: 'f', 'c', 'v', 'l', 'k' or 'w'. These errors appear almost always at the beginning of the word, because correspond to the beginning of a paragraph. They can be solved by finding the way to detect the end of a paragraph. For example, by identifying that the previous character is a '.' (dot).

6. Conclusion

In this paper, we have presented a system for recognizing offline handwritten words. The proposed architecture is inspired in the way that the humans try to identify handwritten words and it is based in the Seq2Seq architecture with the addition of a convolutional network. Our model has three main components oriented to capture the different visual aspects of the handwriting text. First, a convolutional network, that extracts visual features from several image patches, trying to obtain relevant features of the characters present in the word. Second, a Recurrent Neural Network, that captures the sequential relationships among the previous extracted features. Finally, another Recurrent Neural Network, that decodes the sequence of characters to predict the input word.

The main advantage of the proposed model is the integrated management

of local aspects of a word image related with: the characters that compose the word (convolutional network), the sequential horizontal aspects of the word image (encoder) and the sequence of characters in the word (decoder).

The obtained results, when applying our visual model over two standard databases, are competitive with those other published in the literature. We have achieved in some cases the best published results with a direct visual model or with a lexicon search.

As future research we plan to test if this model can also work well with other languages like Chinese or Arabian. Also, we will test the model at line level. With respect to the error analysis, we plan to explore new possibilities by including language models, at character and word levels, in order to improve the decoder component of the model.

Acknowledgements

This work was funded by the Spanish Ministry of Economy and Competitiveness under projects numbers TIN2014-57458-R and TIN2017-85221-R.

References

[1] A. Graves, J. Schmidhuber, Offline handwriting recognition with multidimensional recurrent neural networks, in: Advances in neural information processing systems, 2009, pp. 545–552.

[2] K. M. Sayre, Machine recognition of handwritten words: A project report, Pattern recognition 5 (3) (1973) 213–228.

[3] R. Plamondon, S. N. Srihari, Online and off-line handwriting recognition: a comprehensive survey, IEEE Transactions on pattern analysis and machine intelligence 22 (1) (2000) 63–84.

[4] A. Yuan, G. Bai, L. Jiao, Y. Liu, Offline handwritten english character recognition based on convolutional neural network, in: Document Analysis

Systems (DAS), 2012 10th IAPR International Workshop on, IEEE, 2012, pp. 125–129.

[5] C. N. Manisha, E. S. Reddy, Y. S. Krishna, Role of offline handwritten character recognition system in various applications, International Journal of Computer Applications 135 (2) (2016) 30–33.

[6] J. A. Sánchez, V. Bosch, V. Romero, K. Depuydt, J. de Does, Handwritten text recognition for historical documents in the transcriptorium project, in: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, ACM, 2014, pp. 111–117.

[7] T. Plötz, G. A. Fink, Markov models for offline handwriting recognition: A survey, Int. J. Doc. Anal. Recognit. 12 (4) (2009) 269–298. doi:10.1007/s10032-009-0098-4.
URL http://dx.doi.org/10.1007/s10032-009-0098-4

[8] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE transactions on pattern analysis and machine intelligence 31 (5) (2009) 855–868.

[9] T. Wang, D. J. Wu, A. Coates, A. Y. Ng, End-to-end text recognition with convolutional neural networks, in: Pattern Recognition (ICPR), 2012 21st International Conference on, IEEE, 2012, pp. 3304–3308.

[10] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[11] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, Attention-based models for speech recognition, in: Advances in Neural Information Processing Systems, 2015, pp. 577–585.

[12] J. S. Chung, A. Zisserman, Lip reading in the wild, in: Asian Conference on Computer Vision, Springer, 2016, pp. 87–103.

[13] U.-V. Marti, H. Bunke, The iam-database: an english sentence database for offline handwriting recognition, International Journal on Document Analysis and Recognition 5 (1) (2002) 39–46.

[14] E. Augustin, M. Carré, E. Grosicki, J.-M. Brodin, E. Geoffrois, F. Prêteux, Rimes evaluation campaign for handwritten mail processing, in: International Workshop on Frontiers in Handwriting Recognition (IWFHR'06),, 2006, pp. 231–235.

[15] P. Doetsch, M. Kozielski, H. Ney, Fast and robust training of recurrent neural networks for offline handwriting recognition, in: Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on, IEEE, 2014, pp. 279–284.

[16] M. Kozielski, P. Doetsch, H. Ney, Improvements in rwth's system for offline handwriting recognition, in: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, IEEE, 2013, pp. 935–939.

[17] G. Bideault, L. Mioulet, C. Chatelain, T. Paquet, Spotting handwritten words and regex using a two stage blstm-hmm architecture, in: SPIE/IS&T Electronic Imaging, International Society for Optics and Photonics, 2015, pp. 94020G–94020G.

[18] T. Bluche, J. Louradour, R. Messina, Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention, arXiv preprint arXiv:1604.03286.

[19] T. Bluche, H. Ney, C. Kermorvant, Feature extraction with convolutional neural networks for handwritten word recognition, in: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, IEEE, 2013, pp. 285–289.

[20] A. Graves, S. Fernández, F. Gomez, J. Schmidhuber, Connectionist temporal classification: labelling unsegmented sequence data with recurrent

neural networks, in: Proceedings of the 23rd international conference on Machine learning, ACM, 2006, pp. 369–376.

[21] P. Voigtlaender, P. Doetsch, H. Ney, Handwriting recognition with large multidimensional long short-term memory recurrent neural networks, in: Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on, IEEE, 2016, pp. 228–233.

[22] A. Graves, S. Fernndez, J. Schmidhuber, Multidimensional recurrent neural networks, in: in: Proceedings of the 2007 International Conference on Artificial Neural Networks (ICANN), 2007.

[23] B. Stuner, C. Chatelain, T. Paquet, Cohort of lstm and lexicon verification for handwriting recognition with gigantic lexicon, arXiv preprint arXiv:1612.07528.

[24] M. Sabatelli, Y. Y. Shkarupa, Offline handwriting recognition using lstm recurrent neural networks, in: Proceedings of the 28th Benelux Conference on Artificial Intelligence (BNAIC), 2016, pp. 88–95.

[25] K. Larson, The science of word recognition, Advanced Reading Technology, Microsoft Corporation.

[26] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[27] G. Kim, V. Govindaraju, A lexicon driven approach to handwritten word recognition for real-time applications, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 366–379. doi:10.1109/34.588017.
URL http://dx.doi.org/10.1109/34.588017

[28] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[29] J. Sueiras, V. Ruiz, Á. Sánchez, J. F. Vélez, Using a synthetic character database for training deep learning models applied to offline handwritten recognition, in: International Conference on Intelligent Systems Design and Applications, Springer, 2016, pp. 299–308.

[30] A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: Acoustics, speech and signal processing (icassp), 2013 ieee international conference on, IEEE, 2013, pp. 6645–6649.

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

[32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467.

[33] V. Pham, T. Bluche, C. Kermorvant, J. Louradour, Dropout improves recurrent neural networks for handwriting recognition, in: Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on, IEEE, 2014, pp. 285–290.

[34] S. Johansson, The tagged {LOB} corpus: User\'s manual.

[35] W. N. Francis, H. Kučera, Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers, Brown University, Department of Linguistics, 1989.

[36] T. Bluche, Joint line segmentation and transcription for end-to-end handwritten paragraph recognition, in: Advances in Neural Information Processing Systems, 2016, pp. 838–846.

[37] T. Bluche, Deep neural networks for large vocabulary handwritten text recognition, Ph.D. thesis, Université Paris Sud-Paris XI (2015).

[38] A. Poznanski, L. Wolf, Cnn-n-gram for handwriting word recognition, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[39] J. Almazán, A. Gordo, A. Fornés, E. Valveny, Word spotting and recognition with embedded attributes, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (12) (2014) 2552–2566.

[40] T. Bluche, H. Ney, C. Kermorvant, Tandem hmm with convolutional neural network for handwritten word recognition, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 2390–2394.

[41] E. Grosicki, H. El Abed, Icdar 2009 handwriting recognition competition, in: Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on, IEEE, 2009, pp. 1398–1402.

Biography

Jorge Sueiras received the M.S degree in Statistics and the B.S. degree in computer Science from the Universidad de Oviedo, Spain, in 1995 and 2015 respectively. and is a PhD candidate at the Universidad Rey Juan Carlos, Spain. He led the R&D departments in both Spanish companies, working in industrial applications of machine learning. His research interests are related to machine learning and artificial intelligence.

José F. Velez is an associate professor at Rey Juan Carlos University in Madrid (Spain) where he is a member of the GAVAB research group. His main research interests focus on the developing of Computer Vision Industrial Applications derived from its experience working in research departments of several Spanish Companies.

Victoria Ruiz Parrado She is a PhD student in Rey Juan Carlos University. She received her Bachelor Degree in Mathematics and Statistics from Complutense University in 2013, and her Master Degree in Statistical and Computational Treatment of the Information from Complutense University in 2015. Her research interest lie in areas of Computer Vision and Deep learning. Her scientific contribution in those fields has more to do with handwriting text recognition.

Dr. Angel Sanchez obtained his Bachelor's Degree in Computer Science (1986) and his PhD degree in Computer Science (1990), both from the Technical University of Madrid, Spain. Currently, he is a Full Professor in the School of Computer Science at Rey Juan Carlos University in Madrid (Spain). He has participated in multiple research and technology transfer projects. He also participates regularly as an external evaluator of research projects and has been a member of the program committee in numerous scientific conferences. His research topics cover a broad spectrum within the

Artificial Vision applications, Computer Biometrics, Pattern Recognition and Soft Computing.