Handwriting Recognition by Attribute Embedding and Recurrent Neural Networks

J.Ignacio Toledo, Sounak Dey, Alicia Fornés and Josep Lladós
Computer Vision Center, Computer Science Department,
Universitat Autònoma de Barcelona, Barcelona, Spain
Email: {jitoledo, sdey, afornes, josep}@cvc.uab.es

***Abstract*—Handwriting recognition consists in obtaining the transcription of a text image. Recent word spotting methods based on attribute embedding have shown good performance when recognizing words. However, they are holistic methods in the sense that they recognize the word as a whole (i.e. they find the closest word in the lexicon to the word image). Consequently, these kinds of approaches are not able to deal with out of vocabulary words, which are common in historical manuscripts. Also, they cannot be extended to recognize text lines. In order to address these issues, in this paper we propose a handwriting recognition method that adapts the attribute embedding to sequence learning. Concretely, the method learns the attribute embedding of patches of word images with a convolutional neural network. Then, these embeddings are presented as a sequence to a recurrent neural network that produces the transcription. We obtain promising results even without the use of any kind of dictionary or language model.**

I. Introduction

Offline Handwriting Text Recognition (HTR) is the task of converting a digital image of handwritten text into its textual transcription. First HTR methods were based in the segmentation of individual characters and their posterior recognition using Optical Character Recognition (OCR). But for cursive handwriting, segmentation is a truly difficult problem, and Sayre's Paradox arises; that is, to perform the recognition you need to segment first, but to perform a good segmentation you need to recognize first. In order to tackle this problem, segmentation free methods were proposed such as Hidden Markov Models (HMM) or Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) [10], [26], where the recognition and segmentation was done at the same time. These methods allowed to evolve from isolated character recognition to word and text line recognition. Even more, some recent methods based on attention-models and LSTMs have been proposed to evolve from text line recognition to paragraph recognition, and thus performing a joint transcription and segmentation of text lines [2], [3].

Given the difficulties of HTR, Word Spotting has been raised as an alternative to HTR. Word Spotting [8] is defined as the task of searching words in a document, where the query is a word image (query-by-example) or a text string (query-by-string). Thus, documents are not transcribed, but the information contained can be made accessible in retrieval scenarios.

Lately, a new family of Word Spotting methods have also shown their ability to recognize words. These methods [1], [25] are based on embedding the word image and its transcription into a common attribute space. Then, word spotting consists in finding the nearest neighbors in that space. This approach has been adapted to perform recognition by doing a reversed query-by-example word spotting into a given lexicon. That is, to embed the whole lexicon of words into the attribute space, and then, given a word image, to embed it to find the closest word in the lexicon. This method is indeed performing recognition through word classification.

Recently, this embedding has been integrated into a deep learning architecture, producing impressive results for handwritten word recognition [20]. However, these methods present several disadvantages when compared to traditional HTR. Since they are implicitly performing word classification, the main drawback is the requirement of a lexicon and their inability to deal with out of vocabulary (OOV) words. This might seem a minor drawback for modern languages where huge lexicons are available, but it can be a problem in some scenarios. For instance, in historical documents, the amount of OOV words is usually high, and building a full lexicon might not be feasible. In addition, these methods are recognizing the word as a whole, so, by design, they depend on a good segmentation, and they cannot be extended to text lines.

In this paper, and in order to address these issues, we propose a deep learning HTR method that adapts the attribute embedding to sequence learning. Concretely, we perform the attribute embedding of small pieces of text with a convolutional neural network (PHOCNet) and then we construct a sequence of embeddings that are recognized by Long Short-Term Memory Recurrent Neural Networks with Connectionist Temporal Classification loss (BLSTM+CTC). Therefore, we benefit from the advantages of the attribute embedding, sequence learning and deep learning. As far as we know, this is the first work that attempts to combine both approaches by extending the attribute embedding to sequential recognition.

The rest of the paper is organized as follows. Section 2 presents the system overview, whereas the network architecture is described in Section 3. The experimental results are discussed in Section 4. Section 5 draws the conclusions.

II. System Overview

In this section we describe our two stage approach for handwriting word recognition. The first stage is based in attribute

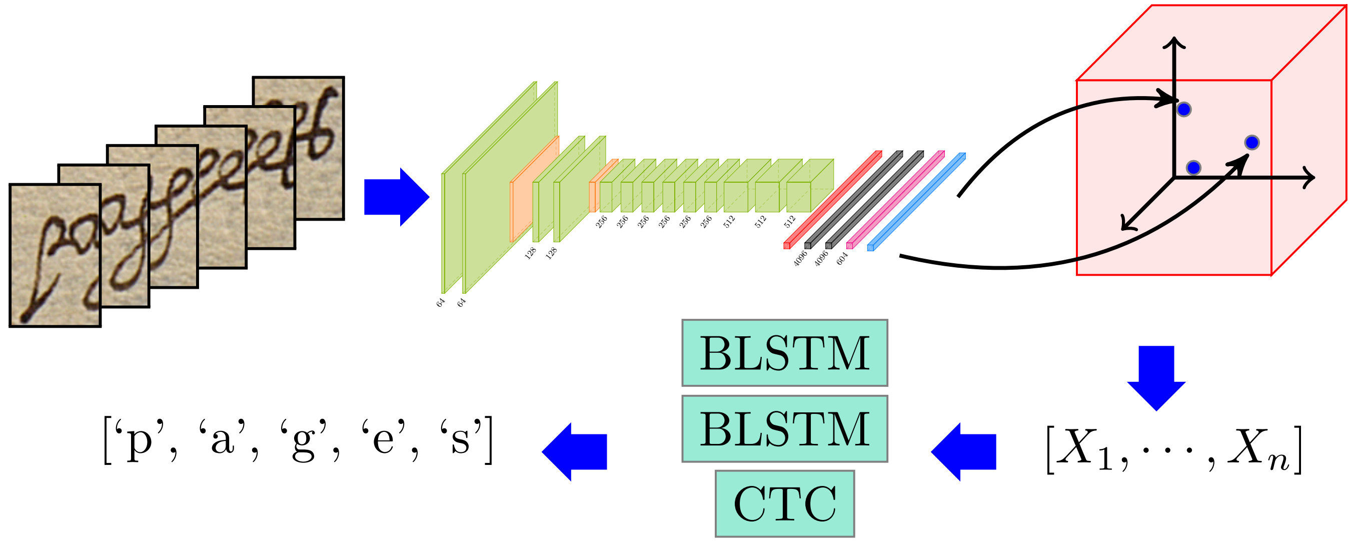

Fig. 1. System architecture. After training a PHOCNet for word attribute embedding, we embed patches of word images into the attribute space. From these points in the attribute space we create a sequence that is passed to a two-layer BLSTM+CTC recurrent neural network that performs the transcription.

embedding, followed by the proper sequence transcription, which is performed over embedded text patches.

A. Attribute Embedding

The Pyramidal Histogram of Characters (PHOC) [1] is used to embed words into an attribute space. In this space, words and word images are characterized by a set of attributes. In the case of PHOC, each attribute represents the presence of a character in a part of the word. Since the position of the characters is also important, the PHOC descriptor is built with a pyramidal structure as follows. In a scenario with n different characters, the first level of the pyramid will have $2n$ dimensions. The first n represent the presence of each character in the first half of the word while the last n represent the presence of a given character in the second half of the word. Each subsequent level of the pyramid will divide the word into smaller portions $1/3$, $1/4$ and $1/5$. The final level of the pyramid contains a selection of the k most common bigrams for that language. The final dimensionality of the PHOC descriptor will be $(2+3+4+5)n+k$.

While the PHOC embedding for text words is trivial, the embedding of word images involves learning. The original approach [1] consists in extracting SIFT features from the word-image, performing a Fischer Vector based clustering and training an individual SVM classifier for each attribute that outputs a likelihood of that word image containing a particular character in a given spatial position. Another possibility for PHOC embedding is to use PHOCNet, a deep convolutional network [25] that is trained to output the PHOC representation of a given word image.

B. Extension to sequences

This kind of attribute embedding has shown to be a reliable representation of words. It has been effectively used for word spotting [1], [25] and recognition [1], [20] by comparing the attribute representation of word images and their transcriptions.

In our case we are interested in the evolution to sequence recognition, towards a lexicon free approach. The key observation is that a word image can be sometimes a prefix or a suffix of another word image, so this attribute embedding approach should also be able to correctly embed smaller patches of words. Then, if we can reliably produce attribute embeddings of arbitrary image patches, that we can extract, for instance, with a sliding window approach, we could use a sequence learning technique in order to learn to transcribe handwriting text.

To test our hypothesis we propose a method based on a modern deep neural network architecture. We start by training a PHOCNet as our attribute embedding choice for word images. Once the training is completed, we do a forward propagation of image patches in this network in order to build a sequence of PHOCs that is then fed to a two layer bidirectional LSTM recurrent neural network with CTC loss. A graphical representation of our proposed architecture can be seen in Fig. 1.

III. Network Architecture

In this section we describe the architecture of the two neural networks that take part in our method. We will discuss the most important characteristics of both the CNN for attribute embedding (PHOCNet) and the Bidirectional Long Short-Term Memory Recurrent Neural Networks with Connectionist Temporal Classification loss that is used for transcription (BLSTM+CTC). We will also provide facts that justify their performance.

A. PHOCNet

The PHOCNet [25] (Fig. 2) is a convolutional neural network architecture (CNN) used for word attribute embedding

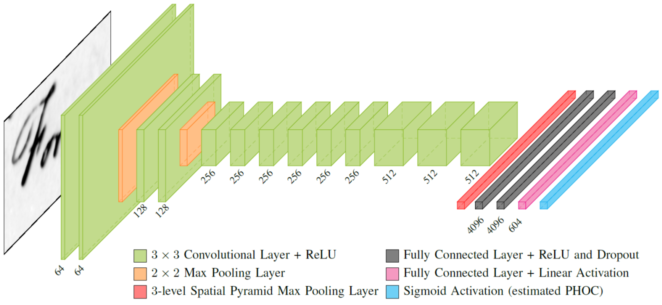
64
64
128
128
256
256
256
256
256
256
512
512
512
4096
4096
604
🟩 3 × 3 Convolutional Layer + ReLU
🟧 2 × 2 Max Pooling Layer
🟥 3-level Spatial Pyramid Max Pooling Layer
⬛ Fully Connected Layer + ReLU and Dropout
🟪 Fully Connected Layer + Linear Activation
🟦 Sigmoid Activation (estimated PHOC)

Fig. 2. The architecture of PHOCNet. Best viewed in electronic format. Extracted from [25].

that has shown impressive results in word spotting. We use the PHOCNet not only because of its good performance but also because it is backed by a carefully thought and theoretically sound design.

Convolutional Neural Networks general layout can be split up in a convolutional and a fully connected part. The convolutional layers can be seen as a feature extractor while the fully connected layers act as a classifier.

Each one of the convolutional layers can be seen as a set of filters that are convolved with its input and followed by a non-linear activation function. These small kernels allow sharing weights for different spatial locations thus considerably reducing the number of parameters and helping in generalization [17]. Convolutional layers are combined with pooling layers in order to introduce a certain amount of translation invariance. In these layers, activations across their receptive field are pooled, and a single activation (usually the one with the maximum value) is forwarded to the next layer [16], [23].

When stacking layers of convolutional and pooling layers, the first layers learn edge detectors that are gradually combined into more abstract features [27]. PHOCNet uses a low number of filters in the lower layers and an increasing number in the higher layers. This leads to the network learning fewer low-level features for smaller receptive fields that gradually combined into more diverse high-level abstract features. Also, all the convolutional layers in PHOCNet utilize filters of size 3x3, since they have shown to achieve better results compared to those with a bigger receptive field as they impose a regularization on the filter kernels [23].

One of the problems that arise when stacking a big amount of layers with traditional activations such as sigmoid or hyperbolic tangent functions is the Vanishing Gradient Problem [18]. This problem has been solved by using Rectified Linear Units (ReLU) as activation function [9]. This function is defined as the truncated linear function $f(x) = \max(0, x)$. By using the ReLU deep CNN architectures became effectively trainable as shown by [16].

The large number of parameters in fully connected layers make them prone to overfitting; even for larger training sets, co-adaptation is a common problem in the fully connected layers [13]. To counter this, various regularization measures have been proposed with Dropout [24] being one of the most prominent. Here, the output of each neuron has a probability (usually 0.5) to be set to 0 during training. Thus, a neuron in a given layer cannot rely on any single specific neuron activation from the preceding layer. This forces the network to learn alternative paths of activations leading to more robust representations and can be seen as an ensemble within the CNN model.

One of the key aspects of PHOCNet is the use of the Spatial Pyramid Pooling (SPP) Layer [12] over the last convolutional layer. This allows the network to accept differently sized input images and output a fixed size representation avoiding the need of a potentially anisotropic rescaling or a cropping. This is crucial when working with word images where cropping is not an option, and, due to the important variability in size and aspect ratio, resizing would result in strong artificial distortions in character shapes and stroke width. In PHOCNet a 3-level Spatial Pyramid max pooling with 4x4,2x2 and 1x1 bin sizes is used. This allows capturing meaningful features at different locations and scales within the word image.

Finally, it is worth mentioning that the network was trained for multi-label classification using the sigmoid activation function in its final layer with cross entropy loss, in contrast to the common single class classification with softmax and categorical cross entropy.

B. BLSTM+CTC

The natural way to deal with sequence learning in neural networks is with Recurrent Neural Networks. In fact, if we consider the resulting network after unfolding for a long sequence, RNNs can be seen as an extreme example of Deep Neural Network. Thus, for RNNs the vanishing gradient problem was known to be a showstopper since the early days of neural networks.

In order to deal with the vanishing gradient problem Long Short Term Memory networks [14] were designed in the late nineties incorporating multiplicative input, output and forget gates. These gates allow the cells to learn to ignore unimportant inputs while keeping their internal state unchanged, and decide when to produce an output, making them specially suited for learning over long sequences.

However, there was still the problem of sequence alignment when the input sequence and the target output were of different lengths. In the late 2000's an algorithm named Connectionist Temporal Classification [11] was invented. By introducing a *blank* "no-output" symbol and a simple algorithm to map network outputs to target sequences and vice-versa, this new algorithm allows the network to perform sequence alignment with differentiable errors. Thus, it allows the training with backpropagation for target sequences with equal or shorter length than the input sequences. Since then, this loss function has been successfully used in tasks like Speech Recognition [11] and Handwriting Recognition [10].

For a better robustness, two LSTM layers, processing the sequence forwards and backwards, are stacked forming a Bidirectional LSTM [10] layer. This kind of bidirectional layer

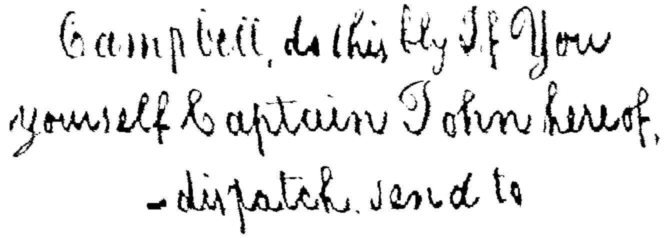

Fig. 3. Some examples of word images in the George Washington dataset. The available images are normalized and binarized.

Fig. 4. Some examples of word images in the Esposalles dataset. We see a high degree of variability both in image size and aspect ratio.

is very useful for offline handwriting recognition where the full sequence is available from the first time-step. In our approach we stack two of these BLSTM layers, resulting in a total of 4 LSTM layers.

However, given the large number of parameters involved in the learning, RNNs are prone to overfitting. Given the success of Dropout [24] for deep neural networks, it is natural to try to use it for RNNs. But, if we use a special architecture like LSTM to keep an internal state for long time, we have to be careful when applying the dropout. One of the first successful attempts to use dropout in RNNs was done in [19] by applying dropout only to the non-recurrent weights. Recent advances in recurrent neural networks [7] allow us to use dropout in all of the connections of an RNNs in a theoretically sound by applying it to the same units at each time step, randomly dropping inputs, outputs, and recurrent connections.

Finally there are also some tricks, discovered empirically, that help to improve the training of LSTM networks [15] like initializing the bias of forget gates to 1 instead of 0 like the rest of the biases.

In our approach we rely on all these advances to design a two layer BLSTM with dropout applied to all its connections, followed by a mandatory Fully Connected layer to match the dimensionality of our output space.

IV. Experiments

In this section we describe the experimental validation of our proposal. We will first explain in detail the datasets used as well as some practical details relative to our training. We will then show and discuss the results.

A. Datasets

For our experiments we used two historical handwritten datasets, both in latin script. Next, we briefly describe them and show some examples below.

1) Washington Dataset: The George Washington (GW) dataset for handwriting recogition [6] is composed of 4894 word images written in 18th century English language with two different writers. The available word images were already normalized to a height of 120 pixels and binarized. We can see several examples of word images in Fig. 3. We use the first of the four different proposed partitions of the dataset which results in 2433 word images for training, 1293 for validation and 1168 for testing.

2) Esposalles Dataset: The Esposalles (BCN) dataset [4], [22] consists of historical handwritten marriages records stored in the archives of Barcelona cathedral. The book was written between 1617 and 1619 by a single writer in old Catalan. The data we used corresponds to 125 pages, with a total of 39527 word images and their transcriptions. The training set contains 100 pages (31501 word images) and the test set contains 25 pages (8026 word images). From the training set, we subtract the last 10 pages (3155 word images) to use them as validation. We can see several examples of word images in Fig. 4.

B. Experimental Setup

For each dataset, we trained the attribute embedding network PHOCNet with the hyperparameter values recommended in [25] for 30.000 iterations. For each experiment, the train, validation and test partitions are the same in all the stages (that is, the PHOCNet and BLSTM+CTC). This means that we train the PHOCNet with and only with the train samples that will later be used when training the BLSTM+CTC.

We used the network to precompute the PHOC of windows of 64 pixel width, with a step size of 8. Each word image was previously padded with 32 pixels of uniform background to the left and to the right to ensure a minimum sequence length for narrower images.

The sequence of precomputed PHOCs was then fed into a recurrent neural network consisting of two bidirectional LSTM layers with 250 neurons with Dropout probability 0.5 in all its connections, followed by a fully connected layer with the required number of neurons for each dataset (82+*blank* for GW and 59+*blank* BCN). The loss function was CTC loss, which requires the extra *blank* symbol.

For training the recurrent neural network, we used early stopping with a tolerance of 30 epochs, and the optimization technique was done with Stochastic Gradient Descent with Nesterov momentum 0.9, a learning rate of $1e^{-4}$ and a decay factor of $1e^{-6}$.

The decoding of the network output was done by selecting the most likely prediction for each timestep and performing the CTC collapse operation, which consists in removing consecutive activations of the same character and the *blank* symbol.

For test evaluation, as well as for the early stopping in the validation set, we use the character error rate (CER) metric.

$$CER = \frac{S + D + I}{N}$$

The CER is calculated as the sum of the character substitutions, insertions and deletions required to transform one string into the other, divided by the total number of characters of the longest string, resulting in a score between 0 and 1 for any pair of words.

C. Results discussion

Table I shows our results. We achieve a CER of 7.32% in the George Washington dataset and an impressive 0.83% in the Esposalles dataset. We observe a significant difference in the performance between both datasets. There are several factors that make these datasets different, starting by the amount of data which is crucial when training with neural networks. There is also the fact that the George Washington dataset is normalized an binarized. We hypothesize that these normalizations might remove some useful information. In fact, for humans, the images from the Esposalles Dataset (Fig. 4) are far more legible than the images from the George Washington dataset (Fig. 3). Finally we would like to remark that the Esposalles dataset consists in marriage records, which usually contain several instances of each unique word. Contrary, the George Washington dataset is a small size free text, and there are only a few instances of each unique word.

Trying to compare our method with other methods is not an straight-forward task. Both works in [1], [20] are based on attribute embeddings. From them, we chose the work from Almazan et al. [1] because it utilizes the original attribute embedding and its source code is publicly available. However the the original method and our approach have important differences, the main one is the requirement of a dictionary. For that reason we consider two different scenarios. In the first (worst case) scenario a lexicon is automatically built from the training examples, thus we will have a minimum lexicon that is always available. In the second (best case) scenario, the transcriptions from the test will also be included in the lexicon, representing a perfect lexicon, so all words can be found in the lexicon. In any case, in both scenarios our method outperforms the original approach by a big margin.

It is also worth noting that when training with the method from [1] only lowercase characters and digits are taken into consideration when building the PHOC representation. Contrary, in our method, all the characters present in the dataset are used, including uppercase, 'ç' and '#' (symbol that denotes crossed out words or characters) symbols for the Esposalles and punctuation marks for the George Washington dataset. Ignoring special characters like '.' or ',' makes the problem easier, whereas transforming all transcriptions to lowercase can help in some examples and damage in others. As a result, the comparison of these methods is even more difficult.

In order to minimize the penalization of the method described in [1] for not using special characters, we removed all punctuation marks from the ground truth, merged the ordinals (1st,2nd) with their corresponding digit and the special character for the initials GW was split into a 'G' and a 'W'. This is a strong simplification, since in the evaluation of our own method, we model and take into account the commas, dots, left and right parenthesis, etc. Even dealing with a much harder task, our method outperforms previous reports by a great margin.

TABLE I
COMPARATIVE WITH OTHER METHODS CER.

Method	Esposalles	GW
Almazan et al. [1] (Train Lexicon)*	6.18%	22.15%
Almazan et al. [1] (Full Lexicon)*	4.28%	17.40%
Fischer [5]	-	≈ 20%
Our approach	**0.83%**	**7.32%**

A more direct comparison can be made with the traditional single layer 100 neuron BLSTM+CTC recurrent neural network like the one described in [5], [10]. This method does not require any kind of language model and trains the network from a sequence of a 9-dimensional handcrafted feature based on statistical and shape information. Results for word-level recognition for the George Washington dataset are reported in a plot in [5] showing a result slightly above the 20% level. In this case we are not certain if punctuation symbols were taken in consideration when calculating the CER.

In case of the Esposalles dataset, the comparison with other existing methods is even more difficult, because most of them perform the recognition at line or record level. The best approach so far is from Romero et al. [21] reporting a WER of 10.1% at line level using a lexicon and a language model. Our method achieves a 2.95% WER in Esposalles without any kind of lexicon or language model but, since we are working at word level, we can not make word insertions or deletions errors but only substitutions. These values, despite not being fully comparable might give a hint of the performance level of our approach.

V. Conclusions

In this paper we present an new handwriting recognition method based on an attribute embedding of patches of word images by a convolutional neural network. Then these embeddings are presented as a sequence to a recurrent neural network that produces the transcription. With this new approach we overcome the limitation of requiring a lexicon that attribute based models have, effectively moving the focus away from word-classification to a real handwriting text recognition. We obtain very competitive results achieving state of the art results in both of the historical handwriting datasets benchmarked.

The most evident future research lines opened by this work are the extension to text lines (something that previous works based on attribute embedding where unable to do by design). In our case this should be possible if we are able to model the white space character between words either at the attribute embedding level or at the RNN sequence transcription level. A second possible improvement would be to make use of language models or lexicon information when available (but not as a requirement of the model). Finally we would also like to perform further experiments with modern handwriting recognition datasets, like the IAM, in order to be easily comparable with other methods in the literature.

Acknowledgment

This work has been partially supported by the Spanish project TIN2015-70924-C2-2-R, the grant 2013-DI-067 from the Secretaria d'Universitats i Recerca del Departament d'Economia i Coneixement de la Generalitat de Catalunya, the Ramon y Cajal Fellowship RYC-2014-16831, and the CERCA Programme/Generalitat de Catalunya. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2552–2566, 2014.

[2] 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.

[3] T. Bluche, J. Louradour, and R. Messina, "Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention," *arXiv preprint arXiv:1604.03286*, 2016.

[4] D. Fernández-Mota, J. Almazán, N. Cirera, A. Fornés, and J. Lladós, "Bh2m: The barcelona historical, handwritten marriages database," in *22nd International Conference on Pattern Recognition (ICPR), 2014*. IEEE, 2014, pp. 256–261.

[5] A. Fischer, "Handwriting recognition in historical documents," Ph.D. dissertation, University of Bern, 2012.

[6] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character hmms," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 934–942, 2012.

[7] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1019–1027.

[8] A. P. Giotis, G. Sfikas, B. Gatos, and C. Nikou, "A survey of document image word spotting techniques," *Pattern Recognition*, 2017.

[9] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[10] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, no. 5, pp. 855–868, 2009.

[11] A. Graves, S. Fernández, and F. Gomez, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *In Proceedings of the International Conference on Machine Learning, ICML 2006*, 2006, pp. 369–376.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 37, no. 9, pp. 1904–1916, 2015.

[13] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," *Journal of Machine Learning Research*, 2015.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 1097–1105.

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network," *NIPS*, pp. 396–404, 1990.

[18] R. Pascanu, T. Mikolov, and Y. Bengio, "On the Difficulty of Training Recurrent Neural Networks," in *International Conference on Machine Learning*, no. 2, 2013, pp. 1310–1318.

[19] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[20] A. Poznanski and L. Wolf, "Cnn-n-gram for handwritingword recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2305–2314.

[21] V. Romero, A. Fornés, E. Vidal, and J. A. Sánchez, "Using the mggi methodology for category-based language modeling in handwritten marriage licenses books," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Oct 2016, pp. 331–336.

[22] V. Romero, A. Fornés, N. Serrano, J. A. SáNchez, A. H. Toselli, V. Frinken, E. Vidal, and J. Lladós, "The esposalles database: An ancient marriage license corpus for off-line handwriting recognition," *Pattern Recognition*, vol. 46, no. 6, pp. 1658–1669, 2013.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[25] S. Sudholt and G. A. Fink, "Phocnet: A deep convolutional neural network for word spotting in handwritten documents," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Oct 2016, pp. 277–282.

[26] P. Voigtlaender, P. Doetsch, and 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.

[27] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *ECCV 2014*, vol. 8689, pp. 818–833, 2014.