Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition

Théodore Bluche and Ronaldo Messina
A2iA SAS
Paris, France
{tb,rm}@a2ia.com

***Abstract*—In this paper, we propose a new neural network architecture for state-of-the-art handwriting recognition, alternative to multi-dimensional long short-term memory (MD-LSTM) recurrent neural networks. The model is based on a convolutional encoder of the input images, and a bidirectional LSTM decoder predicting character sequences. In this paradigm, we aim at producing generic, multilingual and reusable features with the convolutional encoder, leveraging more data for transfer learning. The architecture is also motivated by the need for a fast training on GPUs, and the requirement of a fast decoding on CPUs. The main contribution of this paper lies in the convolutional gates in the encoder, enabling hierarchical context-sensitive feature extraction. The experiments on a large benchmark including seven languages show a consistent and significant improvement of the proposed approach over our previous production systems. We also report state-of-the-art results on line and paragraph level recognition on the IAM and Rimes databases.**

I. Introduction

Since their introduction in 2008, multi-dimensional long short-term memory recurrent neural networks (MDLSTM-RNNs [1]) became established as the state-of-the-art model for handwriting recognition. They were involved in all winning systems in international evaluations [2], [3], [4], and are now a standard component of industrial systems [5], [6]. Yet, they have conceptual drawbacks. Most notably, the features computed by an MDLSTM layer at a given position may depend on quite "distant" intput features. In other words, there is no explicit control of the input context. We observed that this lack of control of context dependency makes the features learnt on text lines images not generalizable to paragraph images. In the scope of handwriting recognition, we think that the context of the whole image is not relevant to predict a signle character. Instead, an area covering the character, and maybe surrounding ones, should be sufficient.

In the wake of the recent developments in deep learning, it would be beneficial to build a network that extracts reusable textual features. A current trend in machine learning is to obtain generic features with a neural network, trained on a lot of data for a well understood task. Such features can then be reused to train new models on different, and sometimes more difficult tasks with less data. Prominent examples can be found in computer vision, where AlexNet or VGGNet are widely used as feature extractors for new tasks with visual inputs. In natural language processing, word embeddings are very popular. One of the goal of this work is to propose a network architecture that could be an AlexNet of document processing. Nowadays, the automatic recognition of handwritten text is no longer the main bottleneck of production systems for document analysis and processing. Their performance is more limited by the line detection and extraction in complex documents, or language identification in multilingual streams. Deep learning opens new opportunities to solve these tasks, and we believe that learnt features that are good for text recognition could represent interesting cues to tackle these problems.

To exploit the large quantity of data to improve the accuracy of models, we need a way to train the networks in days rather than weeks. On the other hand, speed requirements are crucial in production systems. MDLSTMs are not as easily parallelizable as convolutions are, for example, although optimized GPU implementations have been proposed. Bidirectional LSTMs (BLSTMs) are also faster, but require a sequential input. In the literature of neural networks, BLSTMs are often applied to language, and convolutions to images.

Following those observations, we propose a neural network architecture made of a convolutional encoder of the input image and a BLSTM decoder predicting the sequence of characters. Such architectures have been explored in the past [7], [8], [9], [10], [11], but have not yet outperformed the state-of-the-art MDLSTM. MDLSTMs include complex neurons, with an elaborated gating system, designed to control the information flow, and solving the vanishing and exploding gradient issues in recurrent networks. More than merely enabling an efficient recurrence, we believe that the gates are a crucial component in the success of those architectures, maybe more so than the recurrence itself. The main contribution of this paper is the addition of convolutional gates within the convolutional encoder of images.

We evaluated our model on a big multilingual benchmark made of public and private data in seven languages, as well as on standard databases (Rimes [12] and IAM [13]). We show that this new architecture outperforms the state-of-the-art MDLSTM, and allows to extract generic features that are reusable across languages. We also compared the gated convolutions with plain convolutions to highlight the importance of gates in the achieved performance.

The remaining of this paper is divided as follows. Section II exposes the motivations of the proposed system, and presents an overview of the neural network. In Section III we introduce the convolutional gates. The experimental setup is explained

in Section IV, and the results of the experiments are reported in Section V. In Section VI, we propose a discussion of those results and of the model, before concluding in Section VII.

II. Proposed Model

A. Neural Network Overview

Following the observation that convolutional neural networks are widely used in computer vision, and bidirectional LSTM are very popular for language applications, we built a deep neural network that could conceptually be split into three main parts.

The **encoder** of the input image is made of convolutional layers. It processes two-dimensional representations and provides 2D features maps. It contains only about 20% of the model's free parameters but represents the slowest component of the architecture (≈80% of the computation). The goal is to make it as generic as possible to be reusable (e.g. the same for all languages, or for different tasks) in order to factorize the processing time.

The **interface** transforms the 2D image-like representation into the expected 1D representation (we predict sequences of characters).

The **decoder** is a bidirectional LSTM RNN that processes feature sequences to predict sequences of characters. It holds most of the capacity of the network (≈80% of the model's size) but has a fast processing (≈20% of the computational time)

B. Motivations

Several motivations led to the design of the proposed model.

Leveraging research experience from other fields: a lot of research and good results were reported with convolutional neural networks on images (e.g. for object recognition) and LSTMs on language (e.g. speech recognition, machine translation). Since the inputs of our system are images, and the outputs are sentences, it makes sense to use the presented architecture.

Reusable features: The lower layers of a neural network can be interpreted as a learnt feature extractor. In many applications, the lower layers of a network trained for a specific task are used to perform a new task. The problem with LSTMs is the lack of control on the locality of the extracted features, Using convolutions makes the result independant of the big picture (the same information is extracted at a given position, independently of the input being a word, line, paragraph or page image). Since the extracted features are used to recognize the text, they hold some textual-content information and could be useful for language identification, neural document layout analysis, attribute detection, attention models, and so on.

GPU training: MDLSTMs do not lend themselves easily to GPUs (low expected speedup, hard to handle variable-sized inputs, not implemented in most public neural net GPU libraries). If we are to train our models with a lot of data, CPU training will become prohibitive in training time. Finding good, GPU-compatible models was a pre-requisite to the design of the model. Using components like convolutions, linear layers, 1D recurrences was the main constraint

Tradeoff between speed, size, and accuracy: One of the most important concern for production systems is to provide accurate systems, which are also fast and small to enable on-device recognition. Analyzing the models and layers showed that convolutions are faster than LSTMs, and 1D LSTMs are faster than MDLSTM.

C. Multilingual System

Traditionally, we train our systems on language-specific data. For example, for the French neural network, we only use available data in French. For low resource languages, we fine-tune a neural network trained on another language, but still on language specific data. There is however a major issue with that approach: when we do not have much data for one language, it often means we only have one collection of documents. Thus, not only do we adapt the model to a new language, but we also adapt it to a specific collection, which might then be risky in production.

In our model, we first train the network on all available data, in all considered languages. Thus the encoder should extract features that are good across different collections. In a subsequent step, we fine-tune only the decoder to each language. This approach has several benefits. First, the encoder represents most of the computation time. Adapting only the decoder results in a very fast training. Since we keep a shared encoder, it does not specialize to a specific language or collection, so we can hope to have relatively generic features. Moreover, the model is now factorized, as its first part is shared across languages, which saves storing size.

In production setups, it might also be the case that the language is not known in advance. Since we trained a generic network first, it may be used in those incertain situations. Moreover, if the features of the encoder turn out to be generic enough, one may imagine using them for other purposes, such as document layout analysis or the rejection of incorrect segmentations. Once again, the benefit could be a factorization of document processing.

III. Gated Convolutions

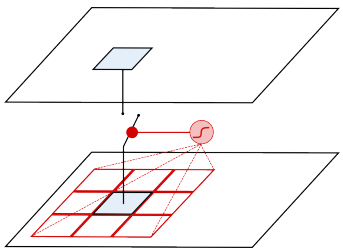

Fig. 1. Convolutional Gate (in red): by applying a convolution filter, followed by a sigmoid activation, the system learns in which context a computed feature is relevant.

In this section, we present the convolutional gates that we use in the encoder. The idea is depicted on Fig. 1. The gate controls the propagation of a feature to the next layer.

Basically, the gate looks at the feature value at a given position, and at neighboring values, and decides whether that feature at that position should be kept or discarded. It allows to compute generic features across the whole image, and to filter when, according to the context, the features are relevant.

The gate (g) is implemented as a convolution layer with sigmoid activation. It is applied to the input feature maps $\mathbf{x}$. The output of the gating mechanism is the pointwise multiplication of the input with the output of the gate:

$$\mathbf{y} = g(\mathbf{x}).\mathbf{x} \tag{1}$$

where

$$\begin{aligned} g(x_{ij}) = \sigma(\quad & w_{00}x_{i-1,j-1} + w_{01}x_{i-1,j} + w_{02}x_{i-1,j+1} + \\ & w_{10}x_{i,j-1} + w_{11}x_{i,j} + w_{12}x_{i,j+1} + \\ & w_{20}x_{i+1,j-1} + w_{21}x_{i+1,j} + w_{22}x_{i+1,j+1}) \end{aligned}$$

which is reminiscent of the input or output gates in LSTMs, except that the context is in the input space rather than coming from outputs of the layer at neighboring positions. In that sense, it is also quite similar to the gating mechanism proposed in Quasi-RNNs [14].

Fig. 1 is actually a simplistic representation of the gate. In fact, the gate computes a value in $[0, 1]$ for each feature (i.e. each dimension of the feature vectors at each position), based on the whole feature vectors at neighboring positions. Thus the context taken into account to decide whether a given value is relevant is relatively rich.

In Fig. 2, we show a real example of what the gate does. In the example, the gate is applied to intermediate feature maps computed from the image of the word *"television"*. We display the feature maps before and after the gate, as well as the maps of gate values.

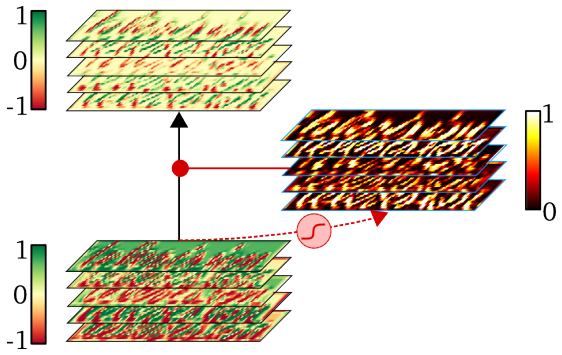

Fig. 2. Visualization of the effect of a convolutional gate.

That example illustrates what, in our opinion, is very interesting with gates. First, we see that there is an effective selection or filtering of features. The gating system allows a feature to be excitatory, inhibitory, or absent, whereas with a tanh it has either a positive or a negative effect, and with a sigmoid it is either present of absent.

IV. Experimental Setup

A. Data

To train our model, we used as much data as possible, from various sources. We did not find any public data for some languages, so we collected data from other sources to build private collections. For several collections, we did not have a line-level ground-truth. We applied methods similar to the one presented in [15] to align page-level transcript to text lines. The obtained dataset may contain errors, since we did not manually validated the obtained data. Overall, we have about 133k text line images across seven languages, including 30% of private data (mainly Russian). The number of text lines in each collection is reported in Table I.

TABLE I
Training data for the multilingual system. Datasets marked with (*) correspond to private data and amount for 30% of the total number of lines. Datasets marked with † were automatically labeled by aligning images with transcripts and contain errors. Datasets marked with ^{a} were augmented with line deformations.

Lang.	Dataset	Num.Lines
English	Maurdor^a	32,475
	IAM	6,482
	IBM†	4,910
	Private.EN*	9,404
French	Rimes	10,532
	Maurdor	26,870
Spanish	Germana	11,505
	Cristo Salvador	971
	Private.SP*	452
German	Private.DE†*	6,939
Italian	Private.IT*	2,261
Portuguese	Private.PT*	870
Russian	Private.RU*	19,312
	TOTAL	**132,983**

B. Architecture Details

We built neural networks according to the principles previously descibed. To optimize different objectives (accuracy, speed and size), we designed two architectures. Both are made of a convolutional encoder, a max-pooling across the vertical dimension and a recurrent decoder. They are depicted on Fig. 3. The encoder consists of a 3x3 convolutional layer with 8 (resp. 4) features, a 2x4 convolutional layer with 16 (resp. 8) features, a 3x3 convolutional gate, a 3x3 convolutional layer with 32 (resp. 16) features, a 3x3 convolutional gate, a 2x4 convolutional layer with 64 (resp. 24) features, a 3x3 convolutional layer with 128 (resp. 32) features, for the big (resp. small) network. The encoder of the small network has an additional 3x3 convolutional layer with 64 features. The decoders are made of 2 bidirectional LSTM layers of 128 (resp. 50 and 100) units, with a linear layer of 128 (resp. 100) neurons in between.

For each architecture, two networks are created: a slow network operating on original images, and a fast network operating on downscaled images to about 70% of their original size. We will refer to these networks as Accurate (A) and

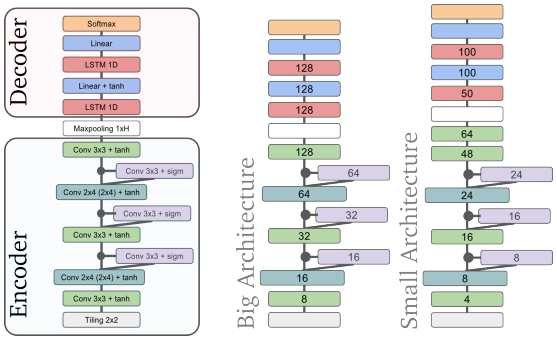

Fig. 3. Architectures of the proposed neural networks.

Fast (F) for the two instances of the big architecture, and FastSmall (S) and FasterSmall (X) for the two instances of the small architecture. Overall, the big architecture contains about 750k parameters and the small one 300k. After weight quantization to 12 bits (which does not hurt the performance), these networks occupy respectively 1.1MB and 500kB of disk space.

C. Training

We trained the network to minimize the Connectionist Temporal Classification (CTC [16]) objective function. We performed the optimization with stochastic gradient descent, using the RMSProp [17] method with a base learning rate of 0.0004 and minibatches of 8 examples. Following the motivations previously exposed, we first train each of the four models on all the available data, to obtain generic Latin-script neural networks. The decoder part is subsequently adapted to each language, using the features extracted with the encoder. The encoder is not fine-tuned to maintain generic, language and collection-independent features. For comparison purposes, we also adapted to whole architecture to each language, and trained neural networks without gates.

V. Results

A. Multilingual Training

In this section, we present the results of the networks trained on all the available data, in all languages. We will refer to these networks as *Generic* models, since they are not sepecific to a language. We compare the four networks (Accurate, Fast, FastSmall and FasterSmall) to the previous version of A2iA TextReader^{TM}.

A2iA TextReader^{TM}comes with two neural networks for each language (an accurate setup and a fast one). Each network is an MDLSTM-RNN following the architecture presented in [1], or variations thereof (e.g. different numbers of neurons in hidden layers, or subsampling convolutions before the first MDLSTM to get faster decoding). In most cases, the accurate version operates on original (300DPI) images, and the fast version on downscaled input images to about 70% of the original size. Each one of these networks is only trained on language specific data, although for low-resourced languages, the initial weights come from the network trained on another language.

The comparison is carried out on a large benchmark including the validation and test sets of all public datasets, mobile-captured pictures of some images from IAM and Maurdor, some private data (not necessarily coming from the same collections as the training sets). For languages with too few data, or for which we had only one collection, we also included synthetic line snippets generated with the method presented in [18].

The character error rates of the neural networks applied alone, without any lexicon or language model are reported in Table II. We compare the results of the previous version of A2iA TextReader^{TM}to the generic gated convolutional recurrent neural network (GCRNN). We see that for all languages except French and Russian, the generic GCRNN is better, and even the smaller and faster architectures are competitive. Note that the comparison here is between seven language-specific networks of the previous version of A2iA TextReader^{TM}and a single GCRNN network.

The Portuguese networks in the previous version of A2iA TextReader^{TM}are actually the networks trained on French data, due to the lack of training documents in Portuguese when the systems were created. They are intended to be applied with a Portuguese language model. That explains why the character error rates of the network alone on Portuguese data is so high.

B. Language-Specific Adaptations

To improve the performance of the models, we subsequently adapt the generic models on language-specific data. As previously mentioned, we believe that keeping a shared encoder brings many advantages, such as ensuring that the features are not collection-specific. However, we need to make sure that we do not loose too much potential accurary by fine-tuning only the decoder. In Table II, we see that adapting only the decoder already brings a lot of improvement, making all models better than their counterpart in the previous version of A2iA TextReader^{TM}, by 10 to 40%. Adapting the whole network is rarely helpful, especially for languages with less data. In these cases, the error rates may even be higher than those obtained when only the decoder is adapted. Those results validate our approach and intuition.

C. Comparison with Plain Convolution

In order to measure the impact of convolutional gates, we compared the GCRNNs of Fig. 3 with versions without gates. To keep the same number of parameters and the same depth of the models, we replaced each gate with a simple feed-forward convolutional layer, with the same hyper-parameters as the one in the gate. For example, the first gate in the big architecture is replaced with a 3x3 convolutional layer with 16 features.

We trained all configurations of the obtained CRNN according to the same scheme applied to Gated CRNNs, i.e. first a generic network, and then an adapted version for each language, in which the decoder only is fine-tuned. The results are displayed in Table II. We see that although the results are

TABLE II

COMPARISON OF THE PROPOSED MODELS WITH THE PREVIOUS VERSION OF A2IA TEXTREADER™ ON A BIG MULTILINGUAL BENCHMARK. TWO ARCHITECTURES AND TWO MODES ARE EVALUATED: ACCURATE (A), FAST (F), FASTSMALL (S) AND FASTERSMALL (X). THE REPORTED RESULTS ARE CHARACTER ERROR RATES OF THE NEURAL NETWORKS ALONE, WITHOUT LEXICON OR LANGUAGE MODEL.

Language	**English**				**French**				**Spannish**				**Italian**			
Model	A	F	S	X	A	F	S	X	A	F	S	X	A	F	S	X
A2iA TextReader™	15.2	16.8	-	-	8.8	10.6	-	-	14.4	12.8	-	-	18.3	28.3	-	-
CRNN																
Generic	16.8	18.5	21.3	22.9	12.2	14.4	17.1	19.1	13.4	16.0	18.8	21.4	16.1	19.3	22.3	24.7
Adapt decoder	15.4	17.2	17.6	19.5	10.6	12.2	12.1	14.3	13.3	15.9	18.8	21.4	16.3	18.5	21.5	24.7
Gated CRNN																
Generic	13.9	15.2	17.7	19.1	9.8	11.3	14.2	15.4	10.2	12.5	15.3	16.5	13.6	15.7	19.0	20.0
Adapt decoder	**12.8**	14.2	15.8	16.8	8.0	9.3	10.7	12.1	**9.1**	12.5	15.3	16.4	**12.9**	14.0	19.0	18.6
Adapt whole	12.8	14.2	15.8	16.8	**7.4**	8.9	10.0	11.2	9.1	12.5	15.3	16.4	15.3	16.6	19.4	18.6

Language	**Portuguese**				**German**				**Russian**			
Model	A	F	S	X	A	F	S	X	A	F	S	X
A2iA TextReader™	42.0	45.1	-	-	27.3	34.6	-	-	15.5	19.9	-	-
CRNN												
Generic	19.1	21.9	24.4	27.0	22.4	24.0	27.6	29.7	24.9	28.4	35.8	35.8
Adapt decoder	17.0	18.4	20.5	23.1	19.3	21.7	24.3	27.5	18.1	20.5	20.7	23.3
Gated CRNN												
Generic	15.4	17.7	21.2	24.0	18.6	20.4	24.4	25.1	20.2	22.6	28.5	30.7
Adapt decoder	**12.7**	15.5	17.1	19.6	**17.0**	18.4	21.4	22.9	14.4	16.9	18.8	20.3
Adapt whole	13.6	16.3	17.1	19.6	18.5	21.1	23.8	27.1	**12.5**	14.3	16.8	18.4

competitive with the previous version of A2iA TextReader™, the error rates are much higher than those achieved with the gated version.

D. Comparison with State-of-the-Art on Public Datasets

In order to compare our approach to the state-of-the-art on public data, we fine-tuned the English Accurate model on the IAM database [13] and the French Accurate model on Rimes [12]. Moreover, we estimated language models (LMs) for both languages. They are hybrid word/character language models (LMs) following the method presented in [19]. The LM for IAM is a word trigram estimated on the LOB[1], Brown and Wellington corpora with a vocabulary of the most frequent 50,000 words. The words of the corpora not in the vocabulary are used to estimate a 7-gram character language model. Both are smoothed with Kneser-Ney [20]. The LM for Rimes is a word 4-gram estimated on the training set ground truth, with a vocabulary of 5,000 words, and a character 5-gram.

TABLE III

LINE-LEVEL RESULTS ON THE IAM AND RIMES DATABASES.

		IAM		**Rimes**	
Model	Params	WER%	CER%	WER%	CER%
GCRNN	725k	10.5	**3.2**	**7.9**	**1.9**
[21]	2.6M	**9.3**	3.5	9.6	2.8
[22]	24M	10.9	4.4	11.2	3.5
[23]	10.7M	12.2	4.7	12.9	4.3
[24]	17M	12.7	4.8	12.1	4.4
[25]	500k	13.6	5.1	12.3	3.3

We obtained state-of-the-art results on IAM and Rimes databases (Table III), with nearly four times less parameters. On Rimes, we outperformed the best published system by about 20%.

We also carried out the paragraph-level experiments, without line segmentation, as proposed in [26]. To do so, we replaced the maxpooling in the networks with the attention model of [26]. The results are reported in Table IV. Compared with the line level results, we almost do not loose any performance at the paragraph level. The word and character error rates are also much better than those reported in [26].

[1]without passages of the development and test sets of IAM

TABLE IV

PARAGRAPH-LEVEL RECOGNITION WITHOUT LINE SEGMENTATION.

Dataset.	**Model**	**WER%**	**CER%**
IAM	Bluche [26]	16.4	6.5
	GCRNN	**10.1**	**3.3**
Rimes	Bluche [26]	12.6	2.9
	GCRNN	**7.9**	**2.2**

VI. DISCUSSION

Not only do we achieve better performance with this new model compared to our baseline, but this architecture is also faster than the MDLSTM architecture despite containing more parameters, as one can notice in Table V.

TABLE V

NUMBER OF PARAMETERS AND AVERAGE PROCESSING TIMES OF DIFFERENT NETWORK ARCHITECTURES

	Model	Params	Decoding time
A2iA TextReader™	Accurate	554k	281 ms/line
	Fast	554k	152 ms/line
GCRNN	Accurate	750k	145 ms/line
	Fast	750k	95 ms/line
	FastSmall	300k	88 ms/line
	FasterSmall	300k	73 ms/line

In the proposed architecture, we have conceptually decoupled an *"image-level"* model made of convolutions and a *"language-level"* model made of recurrent layers. By training the encoder on a big amount of data, in several languages and coming from different collections, we aimed at learning generic textual features, which can be used to recognize

several languages, but also in a transfer learning scenario to learn different tasks.

We have seen that even with no fune-tuning to a specific language, the whole network gives competitive results, and could be applied when the language in the document is not known in advance, or when storage space requirements are small and one model per language do not fit. The fine-tuning of the decoder yields good results in every tested language, allowing to share a significant part of the network across languages. The processing is factorized since all decoders operate with a common encoder. When inspecting the features learnt by the encoder, we observed that most of them were indeed generic textual features. For example, we see in Fig. 4 that one seems to detect accents and *i*-dots while another responds to *n*-like shapes.

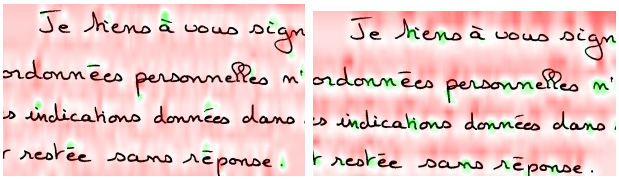

Fig. 4. Example of features extracted by the encoder.

We have carried out several experiments such as language classification and wrong line segmentation rejection from these features and obtained promising results, which are beyond the scope of this paper, but confirm the interest of this approach.

VII. Conclusion

In this paper, we have presented a neural network architecture yielding state-of-the-art results for handwriting recognition. It is made of a convolutional encoder extracting generic features of handwritten text, and an LSTM decoder adapted to each language, predicting character sequences. A crucial aspect of the encoder is the gates, implemented as convolutional layers, which are able to select the relevant features and inhibit the others. In the future, we want to exploit the generic features of the encoder to learn new tasks, such as document layout analysis of language identification.

References

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

[2] A. Tong, M. Przybocki, V. Maergner, and H. El Abed, "NIST 2013 Open Handwriting Recognition and Translation (OpenHaRT13) Evaluation," in *11th IAPR Workshop on Document Analysis Systems (DAS2014)*, 2014.

[3] S. Brunessaux, P. Giroux, B. Grilhères, M. Manta, M. Bodin, K. Choukri, O. Galibert, and J. Kahn, "The Maurdor Project: Improving Automatic Processing of Digital Documents," in *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*. IEEE, 2014, pp. 349–354.

[4] J. A. Sánchez, V. Romero, A. H. Toselli, and E. Vidal, "Icfhr2016 competition on handwritten text recognition on the read dataset," in *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*. IEEE, 2016, pp. 630–635.

[5] B. Moysset, T. Bluche, M. Knibbe, M. F. Benzeghiba, R. Messina, J. Louradour, and C. Kermorvant, "The a2ia multi-lingual text recognition system at the maurdor evaluation," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[6] T. Strauß, T. Grüning, G. Leifert, and R. Labahn, "CITlab ARGUS for historical handwritten documents," 2014.

[7] K. Elagouni, C. Garcia, F. Mamalet, and P. Sébillot, "Text recognition in videos using a recurrent connectionist approach," in *International Conference on Artificial Neural Networks*. Springer, 2012, pp. 172–179.

[8] S. Yousfi, S.-A. Berrani, and C. Garcia, "Deep learning and recurrent connectionist-based approaches for arabic text recognition in videos," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 2015, pp. 1026–1030.

[9] P. He, W. Huang, Y. Qiao, C. C. Loy, and X. Tang, "Reading scene text in deep convolutional sequences," *arXiv preprint arXiv:1506.04395*, 2015.

[10] 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," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[11] C. Adak, B. B. Chaudhuri, and M. Blumenstein, "Offline cursive bengali word recognition using cnns with a recurrent model," in *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*. IEEE, 2016, pp. 429–434.

[12] E. Augustin, M. Carré, E. Grosicki, J.-M. Brodin, E. Geoffrois, and F. Preteux, "RIMES evaluation campaign for handwritten mail processing," in *Proceedings of the Workshop on Frontiers in Handwriting Recognition*, no. 1, 2006.

[13] U.-V. Marti and H. Bunke, "The IAM-database: an English sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2002.

[14] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," *arXiv preprint arXiv:1611.01576*, 2016.

[15] T. Bluche, B. Moysset, and C. Kermorvant, "Automatic line segmentation and ground-truth alignment of handwritten documents," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[16] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *International Conference on Machine learning*, 2006, pp. 369–376.

[17] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, 2012.

[18] P. Krishnan and C. Jawahar, "Generating synthetic data for text recognition," *arXiv preprint arXiv:1608.04224*, 2016.

[19] R. Messina and C. Kermorvant, "Surgenerative finite state transducer n-gram for out-of-vocabulary word recognition," in *International Workshop on Document Analysis Systems (DAS)*, 2014.

[20] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1. IEEE, 1995, pp. 181–184.

[21] P. Voigtlaender, P. Doetsch, and H. Ney, "Handwriting recognition with large multidimensional long short-term memory recurrent neural networks," 2016.

[22] T. Bluche, "Deep neural networks for large vocabulary handwritten text recognition," Ph.D. dissertation, Université Paris Sud-Paris XI, 2015.

[23] P. Doetsch, M. Kozielski, and H. Ney, "Fast and robust training of recurrent neural networks for offline handwriting recognition," pp. 279–284, 2014.

[24] P. Voigtlaender, P. Doetsch, S. Wiesler, R. Schlüter, and H. Ney, "Sequence-discriminative training of recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2100–2104.

[25] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *14th International Conference on Frontiers in Handwriting Recognition (ICFHR2014)*, 2014, pp. 285–290.

[26] 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.