Curriculum Learning for Handwritten Text Line Recognition

Jérôme Louradour and Christopher Kermorvant
A2iA S.A.
39 rue de la Bienfaisance – Paris 75008 France
{jl,ck}@a2ia.com

***Abstract*—Recurrent Neural Networks (RNN) have recently achieved the best performance in off-line Handwriting Text Recognition. At the same time, learning RNN by gradient descent leads to slow convergence, and training times are particularly long when the training database consists of full lines of text. In this paper, we propose an easy way to accelerate stochastic gradient descent in this set-up, and in the general context of learning to recognize sequences. The principle is called Curriculum Learning, or shaping. The idea is to first learn to recognize short sequences before training on all available training sequences. Experiments on three different handwritten text databases (Rimes, IAM, OpenHaRT) show that a simple implementation of this strategy can significantly speed up the training of RNN for Text Recognition, and even significantly improve performance in some cases.**

I. Introduction

The application of interest in this paper is off-line Handwritten Text Recognition (HTR), on images of paper documents. At the time being, the most powerful models for this task are Recurrent Neural Networks (RNN) with several layers of multi-directional Long-Short Term Memory (LSTM) units [1], [2]. Gradient-based optimization of RNN, which cannot be guaranteed to converge to the optimal solution, is a particularly hard issue for two reasons:

First, if we conceptually unfold the recurrences done in the spatial domain (2D, sometimes 1D), we can see RNN as deep models. Because of the numerous non-linear functions that are composed, they are exposed to the burden of exploding and vanishing gradient [3], [4], [5]. In practice, the use of LSTM units, which are carefully designed cells with multiplicative gates to store information over long periods and forget when needed, turned out to be a key ingredient to enable the learning of RNN with standard gradient descent despite the network deepness. There are other ways to efficiently learn RNN, namely enhanced optimization approaches such as second-order methods [6] or good initialization with momentum [7]. These methods are beyond the scope of this paper.

Secondly, RNN are used here for an unconstrained "Temporal Classification" task [8], where the length of the sequence to recognize is in general different from the length of the input sequence. In HTR, the goal is to detect occurrences of characters within a stream of image, without *a priori* segmentation, in other words without knowing the alignment between the pixels and the target characters. So the models must be optimized to solve two problems at the same time: localizing the characters *and* classifying them.

Because of all these aspects, training RNN takes a particularly long time. Here we propose to make the training process more effective by using the concept of Curriculum Learning, that has already been successfully applied in the context of deep models and Stochastic Gradient Descent [9]. The key idea is to guide the training by carefully choosing samples so as to start simple and progressively increase the complexity of training samples. The main motivation is to speed up the learning progression, without any loss of generality in the end. Gradually increasing the complexity of the task has been demonstrated to make learning faster and more robust in several scenarios. This idea has been exploited in classification [9], grammar induction [10], [11], robotics [12], cognitive science [13] and human teaching [14].

In this paper, we show how the Curriculum Learning concept can be naturally be applied to RNN in the context of Handwritten Text Recognition, using the text sequence length as a measure of its complexity. We give empirical evidences that our proposal significantly speeds up the learning progression. The principle is general enough to be applied to any sequence recognition task, and to any kind of model optimized using a gradient-based method.

II. A CURRICULUM FOR TEXT RECOGNITION

A. Two tasks when learning to recognize text: Localization and Classification

In text recognition, locating the characters is necessary to learn to recognize them. However, in many public database for Handwriting Recognition, the positions and the text content are given for each page or paragraph, not for characters. The localization of the lines is reasonably easy to obtain using automatic line segmentation. But locating the characters is a more difficult problem, particularly in the case of handwritten text where even humans can disagree on how to segment the characters. This is why a Connectionist Temporal Classification (CTC) approach as proposed by [8] is a very practical way to train RNN models without intensive labeling effort.

In their CTC approach, [8] efficiently compute and derive a cost function that is the Negative Log-Likelihood (NLL), with the assumptions that all the frame probabilities are independent and that all possible alignments are equiprobable. Besides, an additional label is considered: the *blank*, which stands for "no character" but also for "zone in-between two characters", meaning that the *blank* label can be produced between any two different characters. To the best of our knowledge, taking into account all the possible alignments (including the *blank*) is the most effective approach in training RNN to detect characters. But it also unveils a vague localization of the characters, especially at the beginning of the training process, when the RNN gives quasi-random guesses for the posteriors of the labels (see the CTC Error Signal of Fig.4 in [8]).

Several studies about Text Recognition have revealed that the training process of RNN is particularly long [15]. Not only because of the heavy computational complexity due to the recurrences, but also because the learning progression frequently starts with a plateau. A high number of model parameter updates is needed before the cost function starts to decrease. In some extreme cases, the learning seems to never start, as if the optimization process quickly got stuck in a poor local minimum.

B. Building a suitable Curriculum

One of the reasons for this difficulty to start learning is the fact that when initializing with quasi-random model parameters, the RNN has little chance to produce a reasonable segmentation. Moreover, it is clear that the longer the sequences are, the more serious the problem is. In a nutshell, it is hard and inefficient to learn long sequences at first.

Thinking in the same spirit as [9], let us make a parallel with how to teach kids to read and write. A natural way is to do it step by step: first teach him to recognize characters by showing him isolated symbols, then teach him short words, before introducing longer words and sentences. A similar Curriculum Learning procedure can be done when optimizing neural networks by gradient-descent (*e.g.* RNN using CTC): First optimize on a database of isolated characters (if available), then on a database of isolated words, and finally on a database of lines[1]. But since having access to the positions of characters and/or words may be costly or impossible, we propose here to adapt this proposition to the case where only lines can be robustly extracted from the training database. Keeping in mind that the difficulty when starting to train RNN is related to the length of the training sequences, a general way to build a Curriculum Learning for Text Line Recognition is to first train on short lines, before including long ones.

C. Proposal: continuous curriculum

In practice, it is awkward to build a step-wise schedule by splitting a database with respect to the sequence lengths. Instead, we prefer to handle a probability to draw a sample line from the training database. The idea of defining such a probability for probing the training database has already been successfully applied in Active Learning [16], [17]. If $(\mathbf{X}_t, Y_t)$ denotes a training sample (an image along with the corresponding target sequence of labels), we propose to draw this sample with the following probability parameterized by λ:

$$P_\lambda\left(\text{train on } (\mathbf{X}_t, Y_t)\right) = \frac{1}{N_\lambda}\Big(shortness(\mathbf{X}_t, Y_t)\Big)^\lambda \quad (1)$$

where

- $N_\lambda = \sum_t \left(shortness(\mathbf{X}_t, Y_t)\right)^\lambda$ is a normalization constant so that (1) defines a probability over the set of all the available training samples.
- $shortness \in [0, 1]$ is a bounded value to represent how easy is a training sample. Here it is based on the sequence length, discussed below.
- $\lambda \geq 0$ is an hyper-parameter to tune how much the short words are favoured.

[1]RNN cannot decode paragraphs, just single lines: the common RNN architectures collapse the 2D input image into a 1D signal just before aligning using CTC [1].

The particular setting $\lambda = 0$ amounts to the baseline approach where samples are drawn randomly with flat probability and with replacement. So λ can be tuned during the training process. In our experiments, we start with $\lambda = 3$ and linearly decrease λ until 0, during the equivalent of the first 5 epochs of training. And one epoch is about 10k to 100k different lines of text (see number of labeled lines in table I).

Concerning the $shortness$ measure, we propose to use the following simple form:

$$shortness(\mathbf{X}_t, Y_t) = \frac{1}{\max(m, |Y_t|)} \qquad (2)$$

where $|Y_t|$ is the length of the target sequence (number of characters), and $m > 0$ is a minimal length which stands as a clipping threshold. Using $m = 1$ is needed to avoid numerical problems when there are empty target sequences in the training set. Using more than 1 can be useful to avoid favouring too much on very small words, such as frequent pronouns, or punctuation marks when they are considered as a word with a single character. We used $m = 5$ in our experiments, as it is a common length for short words.

Note that we could use in (2) the width of the input images $|\mathbf{X}_t|$ instead of (or along with) the sequence length $|Y_t|$. It also makes sense and these two measures are actually correlated. But the target length $|Y_t|$ has the advantage of being independent of the resolution of the images, and is also a notion that can be used in other applications than Vision.

III. Experiments

A. Databases

Three notated public handwriting datasets are used to evaluate our system:

- the IAM database, a dataset containing pages of handwritten English text,
- the Rimes database, a dataset of handwritten French letters used in several ICDAR competitions [2].
- The OpenHaRT database, a dataset of handwritten Arabic pages, used in two NIST Open Handwriting Recognition and Translation Evaluation (lastly OpenHaRT 2013).

For all these databases, the localization of the words is available. So we could compare the continuous Curriculum strategy proposed in section II-C with the simple "by-hand" Curriculum which consists in first training RNN to recognize words and secondly to recognize lines.

We use distinct subsets of pages to train and to evaluate RNN models. In the case of the "by-hand" Curriculum, we carefully used the same subset of pages in the training set of words and in the training set of lines. Table I gives the number of data in each training set. The number of different characters depends on the language[2] and also on the punctuation marks that have been labeled in the database. For Arabic recognition, we used a fribidi conversion that map 37 Arabic symbols into 128 different shapes. The resolution of images to feed the network is fixed to 300 dpi. Original OpenHaRT images (*resp.* Rimes images) are in 600 dpi (*resp.* 200 dpi) and they were rescaled with a factor 0.5 (*resp.* 1.5).

			Training subset		
Database	Language	# different characters	# labeled lines	# characters (in lines)	# labeled words
IAM	English	78	9 462	338 904	80 505
Rimes	French	114	11 065	429 099	213 064
OpenHaRT	Arabic	154	91 811	2 267 450	524 196

TABLE I. Number of data in the training sets.

B. Modeling and learning details

The RNN topology we use is the one described in [1] except that the sizes of the filters have been adapted to images in 300 dpi: we used 2x2 input tiling, and 2x4 filters in the two sub-sampling hidden layers (which are convolutional layers without overlap between the filters). The LSTM layers scan the inputs in 4 directions, and the computations can be parallelized over the 4 directions.

All the models were optimized using Stochastic Gradient Descent [18]: a model update happens after each training sample (*i.e.* each line of characters) is visited. The learning rate is constant, and was fixed to 0.001 in all our experiments.

C. Performance assessment

As we are interested in convergence speed, we plot convergence curves that represent the evolution of some costs with respect to a unit of progression of the training algorithm. In our case, we use Stochastic Gradient Descent [18] and the unit of progression could be for instance the number of updates, that is the number of training samples that have been browsed. Given that

[2]for instance there are some diacritics in French that do not exist in English. All tasks here are case-sensitive.

the sequence length of training samples is the measure of complexity here, we chose instead to represent the progression by the total number of targets (characters) that have been browsed. This unit is more representative of the computation time than the number of updates, because the inputs are sequences with variable-length.

We remind that the cost optimized using CTC [8] is the Negative Log-Likelihood (NLL), which can be averaged over the number of training sequences. However, probabilities decrease exponentially with sequence length. For this reason, the NLL average costs are usually higher on databases with long sequences (*e.g.* lines) than on databases with short or middle-length sequences (*e.g.* words). That is why we chose a normalized NLL to monitor the performance of our systems:

$$\text{normNLL}\left(\{(\mathbf{X}_t, Y_t)\}\right) = \frac{\sum_t \text{NLL}(Y_t|\mathbf{X}_t)}{\sum_t |Y_t|} \quad (3)$$

As a relevant but discrete cost to evaluate RNN optical models, we also monitor the Character Error Rate (CER) that is computed by an edit distance, normalized in a similar manner:

$$\text{CER} = \frac{\sum_t EditDistance(Y_t, \widehat{f}(\mathbf{X}_t))}{\sum_t |Y_t|} \quad (4)$$

where $\widehat{f}(\mathbf{X}_t)$ is the sequence recognized by the RNN. The edit distance is the Levenshtein distance.

D. Results and analysis

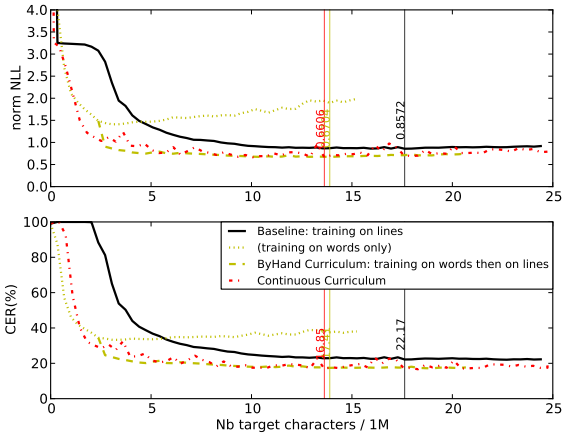

Fig. 1. Convergence Curves on IAM (English).

The convergence curves for learning Handwritten Text Recognition on the three languages, respectively IAM (English), Rimes (French) and OpenHaRT (Arabic), are shown in Figures 1, 2 and 3. They show the

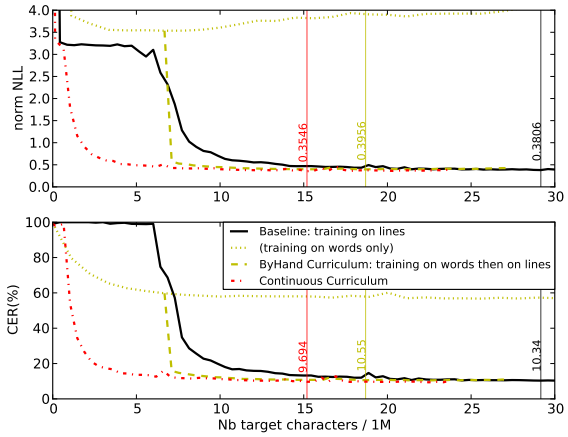

Fig. 2. Convergence Curves on Rimes (French).

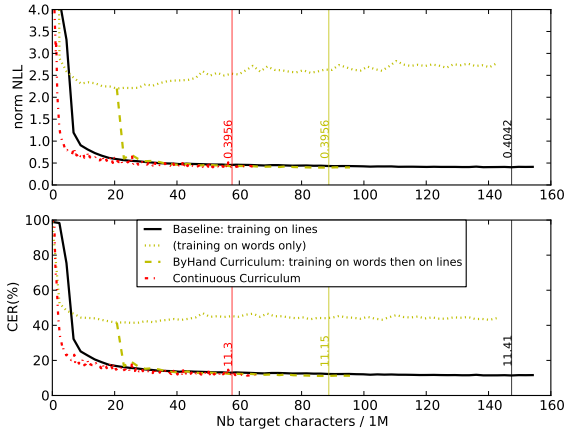

Fig. 3. Convergence Curves on OpenHaRT (Arabic).

progression of the costs on the validation dataset, and the vertical lines point out the best cost values achieved during all the learning process.

All the systems use exactly the same RNN topology and the same optimization procedure, the only difference is the way the training samples are drawn. The baseline system, represented by the black solid curves, consists in shuffling the dataset differently at each epoch.The dashed and dotted yellow curves represent the convergence obtained with a "by-hand" curriculum, starting to train on isolated words, and then training on lines (the switch was done when no more improvement is made by continuing training of words, looking at the performance on the validation set of lines). Finally, the red dashed curve represents the continuous Curriculum approach presented in section II-C.

In the case of the IAM database, a great improvement is achieved by using Curriculum Learning, without any

additional training time: the CER% is decreased from 22% to about 17%. In the case of the Rimes and the OpenHaRT databases, the improvement in performance is slight, but the rate of convergence speed up is remarkable: the whole learning process is roughly twice shorter.

The impact of the Curriculum strategy is visible at the beginning of the training, where the cost functions can be decreased very fast. However, after this initial fast training phase has been completed, and after a transitory phase, the convergence rates are suddenly particularly low whereas the training is not finished. Yet this difficulty to "stop learning" affects all the systems, and indicates that another strategy than the Curriculum should be used to speed up this last learning phase. For instance, techniques to compute a forced alignment [15].

Additional experiments show that, when *adapting* a RNN that has already been trained and that is able to recognize a good part of the characters, the Curriculum strategy does *not* improve over the purely random baseline strategy (neither in performance nor in speed), even in cases where the CER% was high on the new database on which to adapt. This confirms that implementing a Curriculum based on the sequence length can play a crucial role at the beginning of the learning process, but does not affect convergence speed any more after the RNN has learned to detect the positions of the characters.

IV. Conclusion

This paper describes an easy-to-implement strategy to speed up the learning process, that can also provide better performance in the end. The principle is to build a curriculum based on the lengths of the target sequences. Experimental results show that in the case of Recurrent Neural Network for text line recognition optimized by stochastic gradient descent, the first phase of the learning can be drastically shorten, and the generalization performance can be improved, especially when the training set is limited.

At the same time, the slowness of the last phase of the learning remains an issue, that has to be investigated in the future. Further research also includes to experiment our Curriculum Learning procedure in combination with more elaborated optimization methods [6], [7].

Acknowledgments

This work was supported by the French Research Agency under the contract Cognilego ANR 2010-CORD-013.

References

[1] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Proc. NIPS*, 2008.

[2] F. Menasi, J. Louradour, A.-l. Bianne-bernard, and C. Kermorvant, "The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition," in *Document Recognition and Retrieval Conference*, 2012.

[3] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[4] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based*, vol. 6, no. 2, pp. 102–116, 1998.

[5] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *Journal of Machine Learning Research, JMLR*, vol. 28, no. 3, pp. 1310–1318, 2013.

[6] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization," in *Proc. ICML*, 2011.

[7] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. ICML*, 2013.

[8] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.

[9] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. ICML*, 2009.

[10] J. L. Elman, "Learning and development in neural networks: The importance of starting small." *Cognition*, vol. 48, pp. 781–799, 1993.

[11] K. Tu and V. Honavar, "On the utility of curricula in unsupervised learning of probabilistic grammars," in *Proc. IJCAI*, vol. 22, 2011.

[12] T. D. Sanger, "Neural network learning control of robot manipulators using gradually increasing task difficulty," *IEEE Trans. on Robotics and Automation*, vol. 10, 1994.

[13] K. A. Krueger and P. Dayan, "Flexible shaping: how learning in small steps helps," *Cognition*, vol. 110, pp. 380–394, 2009.

[14] F. Khan, X. J. Zhu, and B. Mutlu, "How do humans teach: On curriculum learning and teaching dimension," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., 2011, pp. 1449–1457.

[15] M.-P. Schambach and S. F. Rashid, "Stabilize sequence learning with recurrent neural networks by forced alignment," in *Proc. ICDAR*, 2013.

[16] M. Saar-tsechansky and F. Provost, "Active sampling for class probability estimation and ranking," *Machine Learning*, vol. 54, no. 2, pp. 153–178, 2004.

[17] A. Borisov, E. Tuv, and G. Runger, "Active batch learning with stochastic query-by-forest," *JMLR*, vol. 16, pp. 59–69, 2011.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.