Dropout improves Recurrent Neural Networks for Handwriting Recognition

Vu Pham*†, Théodore Bluche*‡, Christopher Kermorvant*, and Jérôme Louradour*
* A2iA, 39 rue de la Bienfaisance, 75008 - Paris - France
† SUTD, 20 Dover Drive, Singapore
‡LIMSI CNRS, Spoken Language Processing Group, Orsay, France

***Abstract*—Recurrent neural networks (RNNs) with Long Short-Term memory cells currently hold the best known results in unconstrained handwriting recognition. We show that their performance can be greatly improved using *dropout* - a recently proposed regularization method for deep architectures. While previous works showed that dropout gave superior performance in the context of convolutional networks, it had never been applied to RNNs. In our approach, dropout is carefully used in the network so that it does not affect the recurrent connections, hence the power of RNNs in modeling sequences is preserved. Extensive experiments on a broad range of handwritten databases confirm the effectiveness of dropout on deep architectures even when the network mainly consists of recurrent and shared connections.**

***Keywords*—Recurrent Neural Networks, Dropout, Handwriting Recognition**

I. Introduction

Unconstrained offline handwriting recognition is the problem of recognizing long sequences of text when only an image of the text is available. The only constraint in such a setting is that the text is written in a given language. Usually a pre-processing module is used to extract image snippets, each contains one single word or line, which are then fed into the recognizer. A handwriting recognizer, therefore, is in charge of recognizing one single line of text at a time. Generally, such a recognizer should be able to detect the correlation between characters in the sequence, so it has more information about the local context and presumably provides better performance. Readers are referred to [1] for an extensive review of handwriting recognition systems.

Early works typically use a Hidden Markov Model (HMM) [2] or an HMM-neural network hybrid system [3], [4] for the recognizer. However, the hidden states of HMMs follow a first-order Markov chain, hence they cannot handle long-term dependencies in sequences. Moreover, at each time step, HMMs can only select one hidden state, hence an HMM with n hidden states can typically carry only $\log(n)$ bits of information about its dynamics [5].

Recurrent neural networks (RNNs) do not have such limitations and were shown to be very effective in sequence modeling. With their recurrent connections, RNNs can, in principle, store representations of past input events in form of activations, allowing them to model long sequences with complex structures. RNNs are inherently deep in time and can have many layers, both make training parameters a difficult optimization problem. The burden of exploding and vanishing gradient was the reason for the lack of practical applications of RNNs until recently [6], [7].

Lately, an advance in designing RNNs was proposed, namely Long Short-Term Memory (LSTM) cells. LSTM are carefully designed recurrent neurons which gave superior performance in a wide range of sequence modeling problems. In fact, RNNs enhanced by LSTM cells [8] won several important contests [9], [10], [11] and currently hold the best known results in handwriting recognition.

Meanwhile, in the emerging deep learning movement, dropout was used to effectively prevent deep neural networks with lots of parameters from overfitting. It is shown to be effective with deep convolutional networks [12], [13], [14], feed-forward networks [15], [16], [17] but, to the best of our knowledge, has never been applied to RNNs. Moreover, dropout was typically applied only at fully-connected layers [12], [18], even in convolutional networks [13]. In this work, we show that dropout can also be used in RNNs at some certain layers which are not necessarily fully-connected. The choice of applying dropout is carefully made so that it does not affect the recurrent connections, therefore without reducing the ability of RNNs to model long sequences.

Due to the impressive performance of dropout, some extensions of this technique were proposed, including DropConnect [18], Maxout networks [19], and an approximate approach for fast training with dropout [20]. In [18], a theoretical generalization bound of dropout was also derived. In this work, we only consider the original idea of dropout [12].

Section II presents the RNN architecture designed for handwriting recognition. Dropout is then adapted for this architecture as described in Section III. Experimental results are given and analyzed in Section IV, while the last section is dedicated for conclusions.

II. Recurrent neural networks for Handwriting recognition

The recognition system considered in this work is depicted in Fig. 1. The input image is divided into blocks of size 2×2 and fed into four LSTM layers which scan the input in different directions indicated by corresponding arrows. The output of each LSTM layer is separately fed into convolutional layers of 6 features with filter size 2×4. This convolutional layer is applied without overlaping nor biases. It can be seen as a subsampling step, with trainable weights rather than a deterministic subsampling function. The activations of 4 convolutional layers are then summed element-wise and squashed by the hyperbolic tangent (tanh) function. This process is repeated twice with different filter sizes and numbers

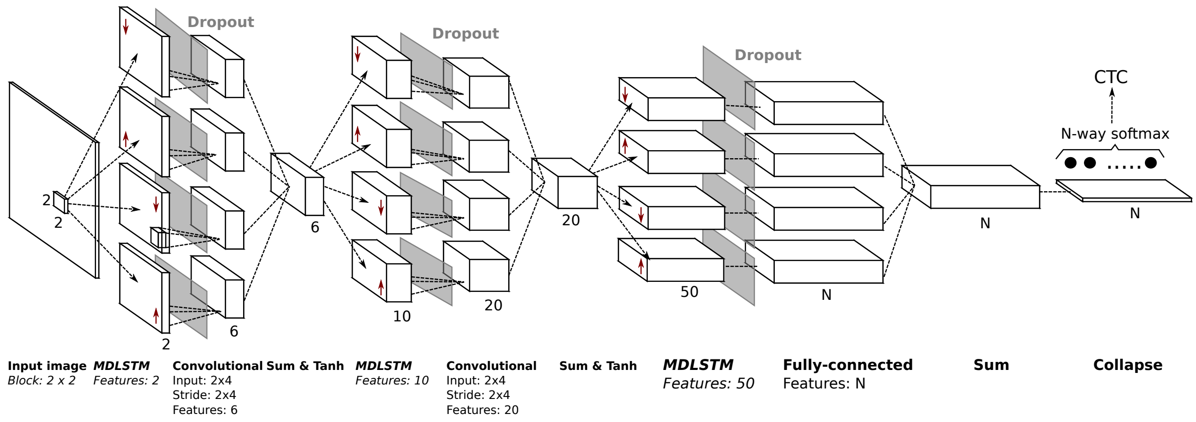

Fig. 1. The Recurrent Neural Network considered in this paper, with the places where dropout can be applied.

of features, and the top-most layer is fully-connected instead of convolutional. The final activations are summed vertically and fed into the softmax layer. The output of softmax is processed by Connectionist Temporal Classification (CTC) [21].

This architecture was proposed in [22], but we have adapted the filter sizes for input images at 300 dpi. There are two key components enabling this architecture to give superior performance:

- *Multidirectional LSTM* layers [23]. LSTM cells are carefully designed recurrent neurons with multiplicative gates to store information over long periods and forget when needed. Four LSTM layers are applied in parallel, each one with a particular scaning direction. In this way the network has the possibility to exploit all available context.
- *CTC* is an elegant approach for computing the Negative Log-likelihood for sequences, so the whole architecture is trainable without having to explicitly align each input image with the corresponding target sequence.

In fact, this architecture was featured in our winning entry of the Arabic handwriting recognition competition OpenHaRT 2013 [11], where such a RNN was used as the optical model in the recognition system. In this paper, we further improve the performance of this optical model using dropout as described in the next section.

III. Dropout for recurrent neural networks

Originally proposed in [12], dropout involves randomly removing some hidden units in a neural network during training but keeping all of them during testing. More formally, consider a layer with d units and let $\mathbf{h}$ be a d-dimensional vector of their activations. When dropout with probability p is applied at this layer, some activations in $\mathbf{h}$ are dropped: $\mathbf{h}^{\text{train}} = \mathbf{m} \odot \mathbf{h}$, where $\odot$ is the element-wise product, and $\mathbf{m}$ is a binary mask vector of size d with each element drawn independently from $m_j \sim \text{Bernoulli}\,(p)$. During testing, all units are retained but their activations are weighted by p: $\mathbf{h}^{\text{test}} = p\mathbf{h}$. Dropout involves a hyper-parameter p, for which a common value is $p = 0.5$.

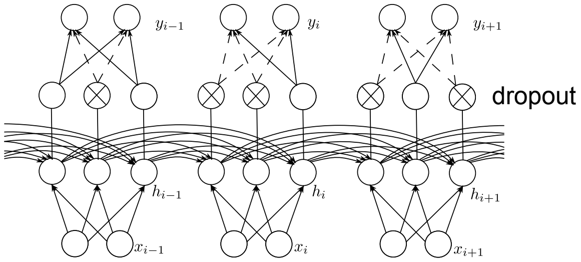

Fig. 2. Dropout is only applied to *feed-forward* connections in RNNs. The *recurrent* connections are kept untouched. This depicts one recurrent layer (h_i) with its inputs (x_i), and an output layer (y_i) which can comprise full or shared connections. The network is unrolled in 3 time steps to clearly show the recurrent connections.

We believe that random dropout should not affect the recurrent connections in order to conserve the ability of RNNs to model sequences. This idea is illustrated in Fig. 2, where dropout is applied only to *feed-forward* connections and not to *recurrent* connections. With this construction, dropout can be seen as a way to combine high-level features learned by recurrent layers. Practically, we implemeted dropout as a separated layer whose output is identical to its input, except at dropped locations ($m_j = 0$). With this implementation, dropout can be used at any stage in a deep architecture, providing more flexibility in designing the network.

Another appealing method similar to dropout is *DropConnect* [18], which drops the connections, instead of the hidden units values. However DropConnect was designed for fully-connected layers, where it makes sense to drop the entries of the weight matrix. In convolutional layers, however, the weights are shared, so there are only a few actual weights. If DropConnect is applied at a convolutional layer with k weights, it can sample at most 2^k different models during training. In contrast, our approach drops the input of convolutional layers. Since the number of inputs is typically much greater than the number of weights in convolutional layers, dropout in our approach samples from a bigger pool of models, and presumably gives superior performance.

In [24], dropout is used to regularize a bi-directional RNN, but the network has only one hidden layer, there are no LSTM cells involved, and there is no detail on how to apply dropout to the RNN. In [14], dropout is used in a convolutional neural network but with a smaller dropout rate because the typical value $p = 0.5$ might slow down the convergence and lead to higher error rate. In this paper, our architecture has both covolutional layers and recurrent layers. The network is significantly deep, and we still find the typical dropout rate $p = 0.5$ yielding superior performance. This improvement can be attributed to the way we keep recurrent connections untouched when applying dropout.

Note that previous works about dropout seem to favor rectified linear units (ReLU) [13] over *tanh* or *sigmoid* for the network nonlinearity since it provides better covergence rate. In our experiments, however, we find out that ReLU can not give good performance in LSTM cells, hence we keep *tanh* for the LSTM cells and *sigmoid* for the gates.

IV. Experiments

A. Experimental setup

Three handwriting datasets are used to evaluate our system: Rimes [25], IAM [26] and OpenHaRT [27] containing handwritten French, English and Arabic text, respectively. We split the databases into disjoint subsets to train, validate and evaluate our models. The size of the selected datasets are given in Table I. All the images used in these experiments consist of either isolated words (Section IV-B) or isolated lines (Section IV-C). They are all scanned at (or scaled to) 300 dpi, and we recall that the network architecture presented in section II is designed to fit with this resolution.

TABLE I. The number of isolated words and lines in the datasets used in this work.

	Rimes		IAM		OpenHaRT	
	words	**lines**	**words**	**lines**	**words**	**lines**
Training	44 197	1 400	80 421	6 482	524 196 [1]	747 676
Validation	7 542	100	16 770	976	57 462	9 525
Evaluation	7 464	100	17 991	2 915	48 308	8 483

[1] For OpenHaRT, only a subset of the full available data was used in the experiments on isolated word.

To assess the performance of our system, we measure the Character Error Rate (CER) and Word Error Rate (WER). The CER is computed by normalizing the total edit distance between every pair of target and recognized sequences of characters (including the white spaces for line recognition). The WER is simply the classification error rate in the case of isolated word recognition, and is a normalized edit distance between sequences of words in the case of line recognition.

The RNN optical models are trained by online stochastic gradient descent with a fixed learning rate of 10^{-3}. The objective function is the Negative Log-Likelihood (NLL) computed by CTC. All the weights are initialized by sampling from a Gaussian distribution with zero mean and a standard deviation of 10^{-2}. A simple early stopping strategy is employed and no other regularization methods than dropout were used. When dropout is enabled, we always use the dropout probability $p = 0.5$.

B. Isolated Word Recognition

1) Dropout at the topmost LSTM layer: In this set of experiments, we first apply dropout at the topmost LSTM layer. Since there are 50 features at this layer, dropout can sample from a great number of networks. Moreover, since the inputs of this layer have smaller sizes than those of lower layers due to subsampling, dropout at this layer will not take too much time during training.

Previous work [28] suggests that dropout is most helpful when the size of the model is relatively big, and the network suffers from overfitting. One way to control the size of the network is to change the number of hidden features in the recurrent layers. While the baseline architecture has 50 features at the topmost layer, we vary it among 30, 50, 100 and 200. All other parameters are kept fixed, the network is then trained with and without dropout.

For each setting and dataset, the model with highest performance on validation set is selected and evaluated on corresponding test set. The results are given in Table II. It can be seen that dropout works very well on IAM and Rimes where it significantly improves the performance by $10 - 20\%$ regardless of the number of topmost hidden units. On OpenHaRT, dropout also helps with 50, 100 or 200 units, but hurts the performance with 30 units, most likely because the model with 30 units is underfitted.

Fig. 3 depicts the convergence curves of various RNN architectures trained on the three datasets when dropout is disabled or enabled. In all experiments, convergence curves show that dropout is very effective in preventing overfitting. When dropout is disabled, the RNNs clearly suffer from overfitting as their NLL on the validation dataset increases after a certain number of iterations. When dropout is enabled, the networks are better regularized and can achieve higher performance on validation set at the end. Especially for OpenHaRT, since its training and validation sets are much larger than IAM and Rimes, 30 hidden units are inadequate and training takes a long time to converge. With 200 units and no dropout, it seems to be overfitted. However when dropout is enabled, 200 units give very good performance.

2) Dropout at multiple layers: Now we explore the possibilities of using dropout also at other layers than the topmost LSTM layer. In our architecture, there are 3 LSTM layers, hence we tried applying dropout at the topmost, the top two and all the three LSTM layers.

Normally when dropout is applied at any layer, we double the number of LSTM units at that layer. This is to keep the same number of active hidden units (on average) when using dropout with $p = 0.5$ as in the baseline where all hidden units are active. We remind that the baseline architecture consists of LSTM layers with 2, 10 and 50 units, so it would correspond to an architecture of 4, 20 and 100 units when dropout is applied at every layer. Since most of free parameters of the networks concentrate at the top layers, doubling the last LSTM layer almost doubles the number of free parameters. Therefore we also have several experiments where we keep the last LSTM layer at 50 units with dropout. Besides, in order to avoid favouring the models trained with dropout because they have greater capacity, we also test those big architectures without dropout.

Their performance are reported in Table III. Since we double the size of LSTM layers, the modeling power of the RNNs is increased. Without dropout, the RNNs with more features at

TABLE II. EVALUATION RESULTS OF WORD RECOGNITION, WITH AND WITHOUT DROPOUT AT THE TOPMOST LSTM HIDDEN LAYER

# topmost LSTM cells	Dropout on top	Rimes CER	Rimes WER	IAM CER	IAM WER	OpenHaRT CER	OpenHaRT WER
30	No	14.72	**42.03**	**20.07**	**42.40**	12.80	37.44
50		15.11	42.62	21.12	43.92	12.89	37.50
100		15.79	44.37	21.87	43.82	**12.48**	**36.50**
200		**14.68**	42.07	22.23	44.83	13.14	37.65
30	Yes	12.33	37.12	18.62	39.61	15.68	43.09
50		**12.17**	**36.03**	**18.45**	39.58	12.87	36.56
100		12.20	**36.03**	18.62	**39.48**	11.50	33.71
200		13.24	38.36	19.72	41.32	**10.97**	**32.64**

Bold numbers indicate the best results obtained for a given database and a given configuration.

TABLE III. EVALUATION RESULTS OF WORD RECOGNITION, WITH DROPOUT AT MULTIPLE LAYERS

# LSTM cells	# LSTM layers with dropout	Rimes CER	Rimes WER	IAM CER	IAM WER	OpenHaRT CER	OpenHaRT WER
2, 10, 50	0 *(no dropout)*	15.11	42.62	21.12	43.92	12.89	37.50
2, 10, 100		15.79	44.37	21.87	43.82	12.48	36.50
2, 20, 50		**13.49**	**39.42**	20.67	42.20	11.32	33.96
2, 20, 100		**13.64**	**39.86**	19.79	**41.22**	11.15	33.55
4, 20, 50		14.48	41.65	**19.67**	**41.15**	**10.93**	**32.84**
4, 20, 100		14.83	42.28	**19.46**	41.47	11.07	33.09
2, 10, 50	1 *(topmost)*	12.17	36.03	18.45	39.58	12.87	36.56
2, 10, 100		12.20	36.03	18.62	39.48	11.50	33.71
2, 20, 50	2 *(top)*	**8.95**	**28.70**	14.52	32.32	10.48	31.45
2, 20, 100		9.29	28.98	15.06	32.96	**9.17**	**28.17**
4, 20, 50	3	**8.62**	**27.01**	**13.92**	**31.48**	11.21	33.11
4, 20, 100		9.98	30.63	**14.02**	**31.44**	9.77	29.58

lower layers generally obtain higher performance. However we observed overfitting on Rimes when we use 4 and 20 features at the lowest LSTM layers. This makes sense because Rimes is the smallest of the three datasets. With dropout, CER and WER decrease by almost 30-40% on a relative basis. We found that dropout at 3 LSTM layers is generally helpful, however the training time is significantly longer both in term of the number of epochs before convergence and the CPU time for each epoch.

C. Line Recognition with Lexical Constraints and Language Modeling

Note that the results presented in Table III can not be directly compared to state-of-the-art results previously published on the same databases [29], [11], since the RNNs only output unconstrained sequences of characters. A complete system for large vocabulary handwriting text recognition includes a lexicon and a language model, which greatly decrease the error rate by inducing lexical constraints and rescoring the hypotheses produced by the optical model.

In order to compare our approach to existing results, we trained again the best RNNs for each database, with and without dropout, on lines of text. The whitespaces in the annotations are also considered as targets for training.

Concretely, we build a hybrid HMM/RNN model. There is a one-state HMM for each label (character, whitespace, and the blank symbol of CTC [21]), which has a transition to itself and an outgoing transition with the same probability. The emission probabilities are obtained by transforming the posterior probabilities given by the RNNs into pseudo-likelihood. Specifically, the posteriors $p(s|x)$ are divided by the priors $p(s)$, scaled by some factor κ : $\dfrac{p(s|x)}{p(s)^\kappa}$, where s is the HMM state, i.e. a character, a blank, or a whitespace, and x is the input. The priors $p(s)$ are estimated on the training set.

We include the lexical contraints (vocabulary and language model) in the decoding phase as a Finite-State Transducer (FST), which is the decoding graph in which we inject the RNN predictions. The method to create an FST that is compatible with the RNN outputs is described in [11]. The whitespaces are treated as an optional word separator in the lexicon. The HMM is also represented as an FST H and is composed with the lexicon FST L, and the language model G.

The final graph HLG is the decoding graph in which we search the best sequence of words $\hat{\mathbf{W}}$

$$\hat{\mathbf{W}} = arg\max_{\mathbf{W}}[\omega \log p(\mathbf{X}|\mathbf{W}) + \log p(\mathbf{W}) + |\mathbf{W}| \log WIP]$$

where $\mathbf{X}$ is the image, $p(\mathbf{X}|\mathbf{W})$ are the pseudo-likelihoods, $p(\mathbf{W})$ is given by the language model, ω and WIP are the optical scaling factor – balancing the importance of the optical model and the language model – and the word insertion penalty. These parameters, along with the prior scaling factor κ, have been tuned independently for each database on its validation set.

For IAM, we applied a 3-gram language model trained on the LOB, Brown and Wellington corpora. The passages of the LOB corpus appearing in the validation and evaluation sets were removed prior to LM training. We limited the vocabulary to the 50k most frequent words. The resulting model has a perplexity of 298 and OOV rate of 4.3% on the validation set (329 and 3.7% on the evaluation set).

For Rimes, we used a vocabulary made of 12k words from the training set. We built a 4-gram language model with modified Kneser-Ney discounting from the training annotations. The language model has a perplexity of 18 and OOV rate of 2.6% on the evaluation set.

For OpenHaRT, we selected a 95k words vocabulary containing all the words of the training set. We trained a 3-gram language model on the training set annotations, with interpolated Kneser-Ney smoothing. The language model has a perplexity of 1162 and OOV rate of 6.8% on the evaluation set.

The results are presented in Tables IV (Rimes), V (IAM) and VI (OpenHaRT). On the first two rows, we present the error rates of the RNNs alone, without any lexical constraint. It can be seen that dropout gives from 7 to 27% relative improvement. The third rows present the error rates when adding lexical constraints without dropout. In this case, only valid sequences of characters are outputed, and the relative improvement in CER over the systems without lexical constraints is more than 40%. On the 4th row, when dropout and lexical constraints are both enabled, dropout achieves 5.7% (Rimes), 19.0% (IAM) and 4.1% (OpenHaRT) relative improvement in CER, and 2.4% (Rimes), 14.5% (IAM) and 3.2% (OpenHaRT) relative improvement in WER. Using a single model and closed vocabulary, our systems outperform the best published results for all databases. Note that on the 5th line of Table V, the system presented in [29] adopts an open-vocabulary approach

TABLE IV. RESULTS ON RIMES

	Valid.		Eval.	
	WER	CER	WER	CER
MDLSTM-RNN	32.6	8.1	35.4	8.9
+ dropout	25.4	5.9	28.5	6.8
+ Vocab&LM	14.0	3.7	12.6	3.5
+ dropout	**13.1**	**3.3**	**12.3**	**3.3**
Messina et al. [30]	-	-	13.3	-
Kozielski et al. [29]	-	-	13.7	4.6
Messina et al. [30]	-	-	14.6	-
Menasri et al. [9]	-	-	15.2	7.2

TABLE V. RESULTS ON IAM

	Valid.		Eval.	
	WER	CER	WER	CER
MDLSTM-RNN	36.5	10.4	43.9	14.4
+ dropout	27.3	7.4	35.1	10.8
+ Vocab&LM	12.1	4.2	15.9	6.3
+ dropout	11.2	3.7	13.6	**5.1**
Kozielski et al. [29]	**9.5**	**2.7**	**13.3**	**5.1**
Kozielski et al. [29]	11.9	3.2	-	-
Espana et al. [31]	19.0	-	22.4	9.8
Graves et al. [32]	-	-	25.9	18.2
Bertolami et al. [33]	26.8	-	32.8	-
Dreuw et al. [34]	22.7	7.7	32.9	12.4

TABLE VI. RESULTS ON OPENHART

	Valid.		Eval.	
	WER	CER	WER	CER
* MDLSTM-RNN	31.0	7.2	34.7	8.4
* + dropout	27.8	6.4	30.3	7.3
+ Vocab&LM	8.3	3.8	18.6	4.9
+ dropout	8.2	3.8	**18.0**	**4.7**
Bluche et al. [11]	-	-	23.3	-
Bluche et al. [11]	-	-	25.0	-
Kozielski et al. [35]	-	-	25.8	10.7

* The error rates in the first 2 lines are computed from the decomposition into presentation forms and are not directly comparable to the remaining of the table.

TABLE VII. NORM OF THE WEIGHTS, FOR DIFFERENTLY TRAINED RNNS.

		Rimes		IAM		OpenHaRT	
		Baseline	Dropout	Baseline	Dropout	Baseline	Dropout
LSTM weights	**L1-norm**	0.162	0.213	0.181	0.220	0.259	0.307
	L2-norm	0.200	0.263	0.225	0.273	0.322	0.382
Classif. weights	**L1-norm**	0.152	0.097	0.188	0.113	0.277	0.175
	L2-norm	0.193	0.120	0.238	0.139	0.353	0.215

The first 2 lines correspond to weights in the topmost LSTM layer (before dropout, if any) and the last 2 lines correspond to classification weights in topmost linear layer (after dropout, if any).

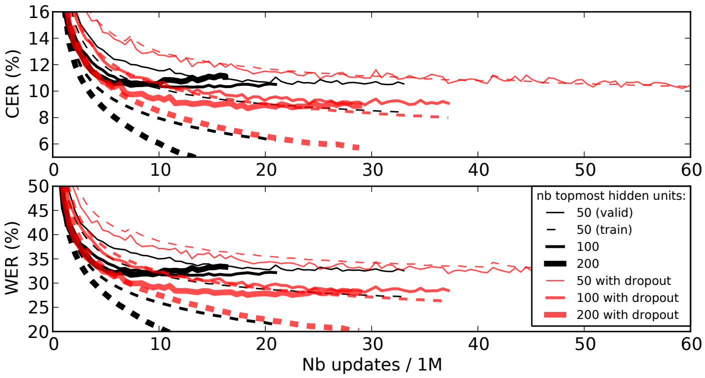

Fig. 3. Convergence Curves on OpenHaRT. Plain (*resp.* dashed) curves show the costs on the validation (*resp.* training) dataset.

and can recognize out-of-vocabulary words, which can not be directly compared to our models.

D. Effects of dropout on the Recurrent Neural Networks

In order to better understand the behaviour of dropout in training RNNs, we analyzed the distribution of the network weights and the intermediate activations. Table VII shows the L1 and L2 norm of the weights of LSTM gates and cells in the topmost LSTM layer (referred to as ”LSTM weights”), and the weights between the topmost LSTM layer and the softmax layer (”Classification weights”). It is noticeable that the classification weights are smaller when dropout is enabled. We did not use any other regularization method, but *dropout seems to have similar regularization effects as L1 or L2 weight decay*. The nice difference is that the hyper-parameter p of dropout is much less tricky to tune than those of weight decay.

On the other hand, the LSTM weights tend to be higher with dropout, and further analysis of the intermediate activations shows that the distribution of LSTM activations have a wider spread. This side effect can be partly explained by the hypothesis that dropout *encourages the units to emit stronger activations*. Since some units were randomly dropped during training, stronger activations might make the units more independently helpful, given the complex contexts of other hidden activations. Furthermore, we checked that the LSTM activations are not saturated under the effect of dropout. Keeping unsaturated activations is particularly important when training RNN, since it ensures that the error gradient can be propagated to learn long-term dependencies.

The regularization effect of dropout is certain when we look into the learning curves given in Fig. 3, where it shows how overfitting can be greatly reduced. The gain of dropout becomes highly significant when the network gets relatively bigger with respect to the dataset.

V. CONCLUSION

We presented how dropout can work with both recurrent and convolutional layers in a deep network architecture. The word recognition networks with dropout at the topmost layer significantly reduces the CER and WER by 10-20%, and the performance can be further improved by 30-40% if dropout is applied at multiple LSTM layers. The experiments on complete line recognition also showed that dropout always improved the error rates, whether the RNNs were used in isolation, or constrained by a lexicon and a language model. We report the best known results on Rimes and OpenHaRT databases. Extensive experiments also provide evidence that dropout behaves similarly to weight decay, but the dropout hyper-parameter is much easier to tune than those of weight decay. It should be noted that although our experiments were conducted on handwritten datasets, the described technique is not limited to handwriting recognition, it can be applied as well in any application of RNNs.

ACKNOWLEDGEMENT

This work was partially funded by the French Grand Emprunt-Investissements d’Avenir program through the PACTE project, and was partly achieved as part of the Quaero Program, funded by OSEO, French State agency for innovation.

References

[1] R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, 2000.

[2] U. Marti and H. Bunke, "Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems," in *Hidden Markov models*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2002, pp. 65–90. [Online]. Available: http://dl.acm.org/citation.cfm?id=505741.505745

[3] S. Marukatat, T. Artires, P. Gallinari, and B. Dorizzi, "Sentence recognition through hybrid neuro-markovian modeling," in *International Conference on Document Analysis and Recognition*, 2001, pp. 731–735.

[4] A. Senior and A. Robinson, "An off-line cursive handwriting recognition system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 309–321, 1998.

[5] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models," *Mach. Learn.*, vol. 29, no. 2-3, pp. 245–273, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1007425814087

[6] 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.

[7] 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.

[8] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," in *Advances in Neural Information Processing Systems*, 2008, pp. 545–552.

[9] F. Menasri, 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.

[10] T. Nion, F. Menasri, J. Louradour, C. Sibade, T. Retornaz, P.-Y. Métaireau, and C. Kermorvant, "Handwritten information extraction from historical census documents," in *International Conference of Document Analysis and Recognition*, 2013.

[11] T. Bluche, J. Louradour, M. Knibbe, B. Moysset, F. Benzeghiba, and C. Kermorvant, "The A2iA arabic handwritten text recognition system at the OpenHaRT2013 evaluation," in *International Workshop on Document Analysis Systems*, 2014 - Accepted.

[12] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[13] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[14] L. Deng, O. Abdel-Hamid, and D. Yu, "A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion," in *International Conference on Acoustics, Speech and Signal Processing*, 2013.

[15] G. Dahl, T. Sainath, and G. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *International Conference on Acoustics, Speech and Signal Processing*, 2013.

[16] J. Li, X. Wang, and B. Xu, "Understanding the dropout strategy and analyzing its effectiveness on lvcsr," in *International Conference on Acoustics, Speech and Signal Processing*, 2013.

[17] M. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *International Conference on Acoustics, Speech and Signal Processing*, 2013.

[18] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning*, 2013.

[19] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *International Conference on Machine Learning*, 2013.

[20] S. I. Wang and C. D. Manning, "Fast dropout training," in *International Conference on Machine Learning*, 2013.

[21] A. Graves, S. Fernandez, 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.

[22] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, L. Bottou, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. MIT Press, 2008, pp. 545–552. [Online]. Available: http://dblp.uni-trier.de/rec/bibtex/conf/nips/GravesS08

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

[24] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *Interspeech*, 2013.

[25] E. Grosicki and H. ElAbed, "ICDAR 2009 handwriting recognition competition," in *International Conference on Document Analysis and Recognition*, 2009.

[26] U. 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. [Online]. Available: http://dx.doi.org/10.1007/s100320200071

[27] NIST, "NIST 2013 Open Handwriting Recognition and Translation Evaluation Plan," 2013. [Online]. Available: http://www.nist.gov/itl/iad/mig/upload/OpenHaRT2013_EvalPlan_v1-7.pdf

[28] G. Hinton and G. Dahl, "Dropout: A simple and effective way to improve neural networks," in *Advances in Neural Information Processing Systems*, 2012. [Online]. Available: http://videolectures.net/nips2012_hinton_networks/

[29] M. Kozielski, P. Doetsch, and H. Ney, "Improvements in RWTH's system for off-line handwriting recognition," in *International Conference on Document Analysis and Recognition*, 2013.

[30] R. Messina and C. Kermorvant, "Surgenerative Finite State Transducer n-gram for Out-Of-Vocabulary Word Recognition," in *11th IAPR Workshop on Document Analysis Systems (DAS2014)*, 2014 - Accepted.

[31] S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, 2010.

[32] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–68, May 2009.

[33] R. Bertolami and H. Bunke, "Hidden Markov Model Based Ensemble Methods for Offline Handwritten Text Line Recognition," *Pattern Recognition*, 2008.

[34] P. Dreuw, P. Doetsch, C. Plahl, and H. Ney, "Hierarchical Hybrid MLP/HMM or rather MLP Features for a Discriminatively Trained Gaussian HMM: A Comparison for Offline Handwriting Recognition," in *International Conference on Image Processing*, 2011.

[35] M. Kozielski, P. Doetsch, M. Hamdani, and H. Ney, "Multilingual off-line handwriting recognition in real-world images," in *International Workshop on Document Analysis Systems*, Tours, Loire Valley, France, Apr. 2014.