Where to apply dropout in recurrent neural networks for handwriting recognition?

Conference Paper · August 2015

DOI: 10.1109/ICDAR.2015.7333848

3 authors, including:

Christopher Kermorvant
TEKLIA
58 PUBLICATIONS **735** CITATIONS
SEE PROFILE

Jérôme Louradour
36 PUBLICATIONS **1,198** CITATIONS
SEE PROFILE

Where to Apply Dropout in Recurrent Neural Networks for Handwriting Recognition?

Théodore Bluche*†, Christopher Kermorvant†‡, and Jérôme Louradour†
*LIMSI CNRS, Spoken Language Processing Group, Orsay, France
†A2iA SA, Paris, France
‡Teklia SAS, Paris, France

***Abstract*—The dropout technique is a data-driven regularization method for neural networks. It consists in randomly setting some activations from a given hidden layer to zero during training. Repeating the procedure for each training example, it is equivalent to sample a network from an exponential number of architectures that share weights. The goal of dropout is to prevent feature detectors to rely on each other. Dropout has successfully been applied to Deep MLPs and to convolutional neural networks, for various tasks of Speech Recognition and Computer Vision. We recently proposed a way to use dropout in MDLSTM-RNNs for handwritten word and line recognition. In this paper, we show that further improvement can be achieved by implementing dropout differently, more specifically by applying it at better positions relative to the LSTM units.**

I. Introduction

The deep neural network proposed by [1] won the ImageNet evaluation introducing an impressive gap in accuracy with respect to previous state-of-the-art systems. This success has provoked a massive interest for all the tricks used to train the classifier neural network, including the Dropout trick.

Dropout is a powerful regularization method originally proposed in [1] for Multi-Layer Perceptrons (MLPs) trained by Stochastic Gradient Descent (SGD). It consists in adding noise to the intermediate data representations in a neural network during the forward propagation step of the SGD, namely by setting to zero a random subset of the hidden activations (right after the application of a nonlinear function, such as a tanh or a rectified linear unit [2]). The backward propagation in the SGD is modified accordingly. Dropout was conceived to prevent the co-adaptation of learned hidden features which is a source of overfitting. It can be seen as an economical approximation to combining a large number of neural subnetworks, as demonstrated mathematically under some particular conditions by [3]. There is now strong empirical evidence that dropout is an efficient data-driven regularizer, meaning that it improves generalization performance on various tasks, and that it has the same effect as weight decay on outgoing weights [4] while having an extremely simple implementation.

Dropout was first experimented with full standard connections within neural networks. A false rumour stated that dropout in combination with parameterized convolutions should not work well, but some attempts demonstrated that it actually is a source of improvement [5]. Then, some works demonstrated that dropout can significantly increase the generalization capacity in architectures with recurrent layers [6]. In particular, [4] and [7] successfully implemented dropout in Recurrent Neural Networks (RNN) composed of Long Short-Term Memory (LSTM) cells [8]: in [4] for Handwriting Text Recognition, and in [7] for Language Modeling and Speech Recognition. As a matter of fact, LSTM networks are now the state-of-the-art in offline Text Recognition [9], [10], which is the application of interest in the current work. In [4], dropout is applied after all the recurrent connections have been computed on the whole input sequence. In [7], it is applied between every two LSTM layers. In both cases, the recurrent connections are left untouched. The underlying motivation for NOT doing dropout within or before the LSTM cells is to prevent threatening the memorization ability of these LSTM cells.

In this paper, we carry out extensive experiments to test where is the best place for dropout in an LSTM network, without any theoretically-based *a priori* on where it should be. Empirical results show that, surprisingly, the best place for dropout is not the one intuitively chosen by [4] and [7].

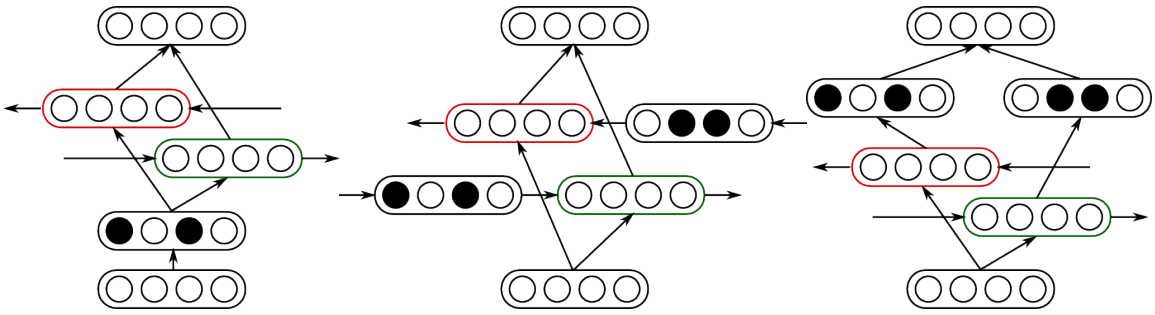
II. Different places where dropout can be applied

In [4] and [7], dropout is used only outside the LSTM layers, so as not to affect the recurrent connections. We can theoretically apply dropout between any two layers, provided that it occurs after a non-linearity and before the application of a parameterized linear transformation.

Figure 1 describes the different places where dropout can be applied, with respect to the LSTM layer. The dropout is represented by layers with black units (circles) inside, that stand for the randomly chosen neurons to disable. LSTM layers are the ones with horizontal arrows that stand for the recurrence (*from-left-to-right* and *from-right-to-left*). There are three different and complementary ways of applying dropout

(a) *Before:* on all the input values of the LSTM.
(b) *Inside:* on the output values of the LSTM within the recurrence loop (*i.e.* dropout is applied on the output at step t before it is used to compute the output at step $t+1$.
(c) *After:* on the output values of the LSTM after all the recurrence iterations have been achieved, without modifying the internal representation inside the LSTM layer.

For the position before the LSTM layer (Figure 1(a)), there are actually two possibilities. The first one is to drop the same LSTM inputs for both directions. The other option is to sample a different set of inputs to drop for the *from-left-to-right* LSTM and the *from-right-to-left* LSTM. Preliminary



(a) Before (b) Inside (c) After

Fig. 1: Place of dropout in RNNs with respect to the LSTM layer.

experiments showed that empirical results are roughly the same, and we present results with the first option.

III. Experimental Setup

A. Databases

TABLE I: Number of pages, lines, words and characters in each dataset

Database	Set	#Pages	#Lines	#Words	#Characters
Rimes *(French)*	*Train*	1,391	10,203	73,822	460,201
	Dev.	149	1,130	8,380	51,924
	Eval.	100	778	5,639	35,286
IAM *(English)*	*Train*	747	6,482	55,081	287,727
	Dev.	116	976	8,895	43,050
	Eval.	336	2,915	25,920	128,531
Bentham *(English)*	*Train*	350	9,198	76,707	419,764
	Dev.	50	1,415	11,580	64,070
	Eval.	50	860	7,868	40,231

We performed the experiments on three public databases, presented on Table I. The *Rimes database* [11] consists of images of handwritten letters from simulated French mail. We followed the setup of the ICDAR 2011 competition. The available data are a training set of 1,500 paragraphs, manually extracted from the images, and an evaluation set of 100 paragraphs. We held out the last 149 paragraphs (approximately 10%) of the training set as a validation set, and trained the systems on the remaining 1,391 paragraphs.

The *IAM database* [12] consists of images of handwritten pages. They correspond to English texts extracted from the LOB corpus, copied by different writers. The database is split into 747 images for training, 116 for validation, and 336 for evaluation.

The last database contains images of personal notes of the British philosopher *Jeremy Bentham*, written by himself and his staff in English, around the 18th and 19th centuries. The data were prepared by the University College, London, during the tranScriptorium project[1]. We followed the setup of the HTRtS competition [13]. The training set consists of 350 pages. The validation set comprises 50 images, and the test set 33 pages.

[1] http://transcriptorium.eu/

All images have an original resolution of 300 DPI, and the average character width, measured from the line images and transcripts, are 38, 39 and 33px for Rimes, IAM and Bentham databases, respectively.

B. Preprocessing

The systems are trained from the annotated text lines. The lines are deslanted using the approach described in [14]. The contrast is enhanced by mapping the 5% darkest pixels to black and 70% lightest ones to white with a linear interpolation in between. Twenty white pixels are added on left and right to account for empty context. All line images were normalized in height to 72px, by rescaling each of the ascenders, descenders and core baseline zones to 24px.

C. Feature Extraction

We used two kinds of features separately: handcrafted features, and raw pixel intensities. In both cases, a sliding window is scanned through the pre-processed image of each line to extract fixed-size vectors of features.

The handcrafted features are geometrical and statistical features presented in [15]: pixel densities, foreground/background transitions, position of the center of gravity, counts of pixel configurations, plus their derivatives. All these features are extracted with a sliding window of width 3px and shift 3px, resulting in 56-dimensional feature vectors.

The "pixel" features are extracted with a sliding window of width 45px (Rimes and IAM) or 57px (Bentham) and shift 3px, and are then rescaled to 20x32px (25x30px for Bentham). The pixel values are normalized to have zero mean and unit variance on the training dataset. A padding of the image was performed before the extraction of pixel values so as to have the same number of vectors for pixel and handcrafted features.

D. Recurrent Neural Networks

For the two types of input features (handcrafted and pixels), we trained Bidirectional LSTM-RNNs. The network architectures considered in this paper alternate LSTM layers with recurrent connections in the two directions [8] (*from-left-to-right* and *from-right-to-left*), and feedforward *tanh* layers. The number of neurons in the hidden layers is 200 units in

the 3 LSTM layers in each direction and in the 2 hidden feed-forward layers. The RNN has one output for each character, plus one non-character output, as defined in [16].

We trained the RNN with the Connectionist Temporal Classification (CTC) objective [16], using Stochastic Gradient Descent and a learning rate of 0.001, to minimize the sequence Negative Log-Likelihood (NLL). After each epoch the NLL loss function is computed on the validation data. The training procedure stops if this cost did not decrease for 20 epochs, and the model producing the lowest validation NLL is kept.

E. Decoding with a Language Model

The decoding is performed at line level with the Kaldi toolkit [17]. It is based on a beam search in a weighted Finite-State Transducer comprising the HMM, lexicon and LM, as described in [18]. It is a hybrid setup, where the optical score is given by the NN posteriors $p(s|x)$, divided by the scaled state priors $p(s)^{\kappa}$, where κ is a tunable parameter (0.5 produced the best results on the validation sets). The RNNs predict characters, so we built single-state character HMMs with a self-loop and uniform transition probabilities.

For Rimes, we built a 4-gram language model with modified Kneser-Ney discounting from the training annotations. The vocabulary is made of 5k words. The language model has a perplexity of 18 and out-of-vocabulary (OOV) rate of 2.9% on the validation set (18 and 2.6% on the evaluation set). For IAM, we used a 3-gram language model limited to the 50k most frequent words, 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. 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 Bentham, we built a 4-gram language model trained on the annotations of the training set, with a special treatment of hyphenation, as described in [19]. The vocabulary includes 7k words plus their possible hyphenations. The OOV rate on the validation set is 5.5% and the perplexity is 98.

IV. Experimental Results

A. Dropout at Different Positions in standalone RNNs without Lexical Constraints

In these experiments, we study the effect of dropout at different relative positions with respect to the recurrences, in different layers in isolation. We compare the effect of the position of dropout with respect to the LSTM layer (before, inside or after, see Figure 1), and of the layers in which it is applied (none, the first one, the second one, the third one, or all the layers).

The experiments were carried out on Rimes, IAM and Bentham. The Character Error Rate obtained using the RNN without Language Modeling (RNN-CER%) are reported in Table II. Bold numbers are the best results for dropout at a single position in a single layer, in all layers, and at all positions in a single layer. The few configurations for which the error rate did not decrease, or increased, are signaled in italics.

TABLE II: Effect of dropout at different positions (RNN-CER%).

	Layers with dropout	**Place of dropout w.r.t LSTM**			
		before	inside	after	everywhere
Rimes *(Features)*	none	8.2			
	1st	6.8	6.7	*8.2*	**6.7**
	2nd	**6.6**	7.3	7.2	6.8
	3rd	7.4	*8.6*	8.0	*8.8*
	all (1+2+3)	**5.0**	5.4	6.8	7.1
(Pixels)	none	9.7			
	1st	**7.1**	7.6	8.1	**7.2**
	2nd	7.5	9.6	9.0	8.8
	3rd	8.0	9.2	7.8	9.1
	all (1+2+3)	**5.8**	6.0	6.5	8.0
IAM *(Features)*	none	10.4			
	1st	9.1	**8.5**	9.8	8.8
	2nd	8.9	9.1	8.6	**8.7**
	3rd	9.1	10.2	9.5	*10.4*
	all (1+2+3)	**7.9**	7.0	9.0	9.4
(Pixels)	none	13.2			
	1st	10.0	**9.1**	11.4	**10.1**
	2nd	10.1	11.1	10.6	10.8
	3rd	10.9	12.3	11.1	12.6
	all (1+2+3)	8.6	**8.4**	10.1	11.4
Bentham *(Features)*	none	11.0			
	1st	**8.5**	9.9	12.3	**8.8**
	2nd	9.8	9.9	10.4	10.0
	3rd	10.5	*11.2*	10.7	*12.3*
	all (1+2+3)	**7.4**	8.1	10.0	8.5
(Pixels)	none	14.0			
	1st	10.4	**9.9**	13.4	**9.7**
	2nd	11.0	13.6	12.2	13.0
	3rd	12.0	*15.1*	12.7	*14.4*
	all (1+2+3)	**8.0**	9.4	10.8	12.3

Besides the fact that dropout almost always helps, we can draw several conclusions. When dropout is **only applied at one position**:

- it is generally better in lower layers of the RNNs, rather than in the top LSTMs, except when it is after the LSTM.
- it is almost always better before the LSTM layer than inside or after it, and better after than inside, except for the bottom layer.

When it is **applied in all layers** (bottom, middle and top):

- among all relative positions to the LSTM, when dropout is applied to every LSTM, placing it after was the worst choice in all six configurations
- before LSTMs seems to be the best choice for Rimes and Bentham, and inside LSTMs is better for IAM.

Moreover, adding dropout **before, inside and after the LSTM** at the same time is not as good as choosing the right position.

On Figure 2, we display the weights of the connections between the inputs and the first LSTM layer, including the gates, without and with dropout at different positions, on Rimes database, with pixel inputs. The recurrent connections are not shown. We notice that the filters are generally sharper with dropout, as if this regularization technique improved the selectivity of the cells, making them more focused on elementary features. This is especially visible for the weights of the gates. Note that the block artefacts that appear in the filters of the RNN without dropout is certainly caused by

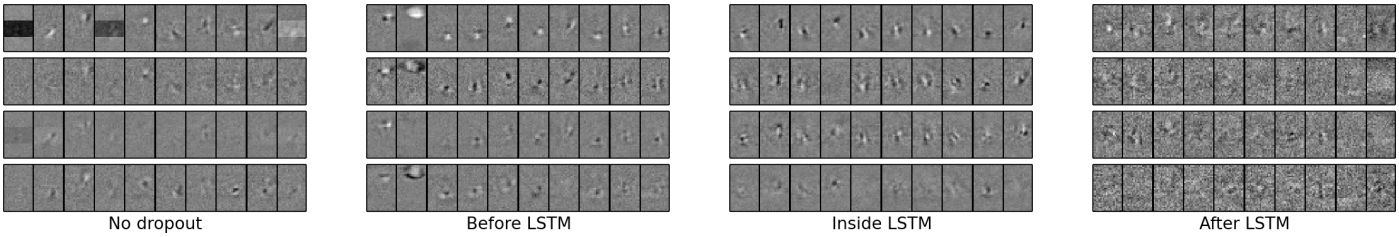


Fig. 2: Visualization of the weights of the connections between the pixel inputs and the first LSTM, with different dropout strategies, for RNNs trained on pixel frames of Rimes database. The first line of each configuration corresponds to the weights of the cell input in LSTMs, and the bottom three lines are the corresponding weights of the input, forget, and output gate.

our three-zone height normalization. Some units seem to have specialized in the detection of the presence or absence of black pixels anywhere in the core region, which is generally sufficient to detect whitespace, for example, ignoring the possible descenders of the line above, or ascenders of the line below.

The values of neighboring pixels are highly correlated. If the model can always access one pixel, it might be sufficient to infer the values of neighboring ones, and the weights will be used to model more complicated correlations. However, with dropout on the inputs, the local correlations are less visible. With half the pixels missing, the model cannot rely one regularities in the input signal, and should model them to make the most of each pixel.

B. Dropout in RNNs with a Language Model

On Table III, we report the results of different dropout strategies in the complete systems including the language models. Dropout is applied in every LSTM layer, at the different positions relative to it. The baselines are RNNs without dropout, and with dropout after LSTM, as proposed by [4].

TABLE III: Effect of dropout at different places in complete Recognition Systems (RNN+LM). CERs inside parentheses correspond to Character Error Rates of the RNN alone.

Database	Place of dropout	Handcrafted Features		Pixels	
		WER	CER (RNN)	WER	CER (RNN)
Rimes	no dropout	12.9	3.7 (8.2)	15.5	4.8 (9.7)
	after	**12.8**	**3.6** (6.8)	**13.3**	4.1 (6.5)
	inside	13.2	3.8 (5.4)	14.3	4.6 (6.0)
	before	13.1	3.7 (**5.0**)	13.8	**4.0** (**5.8**)
IAM	no dropout	11.7	4.0 (10.4)	14.7	5.7 (13.2)
	after	11.8	4.1 (9.0)	13.2	4.7 (10.1)
	inside	**11.6**	**3.9** (**7.0**)	13.3	5.0 (**8.4**)
	before	12.3	4.2 (7.9)	**12.4**	**4.5** (8.6)
Benth.	no dropout	18.1	7.0 (11.0)	21.3	9.0 (14.0)
	after	17.3	6.9 (10.0)	19.1	7.7 (10.8)
	inside	17.7	6.8 (8.1)	20.0	8.5 (9.4)
	before	**16.6**	**6.2** (**7.4**)	**17.8**	**6.9** (**8.0**)

We observe that for Rimes, the best results are achieved with dropout after LSTMs, despite the superior performance of dropout before for the RNN alone. For IAM, dropout inside LSTMs is only slightly better. The main difference between the RNN alone and the complete system is that the former only considers the best character hypothesis at each timestep, whereas the latter potentially considers all predictions in the search of the best transcription with lexical constraints.

Therefore, applying dropout after the LSTM in the top layer(s) might be beneficial for the beam search in the decoding with complete systems (cf. Sect. III-E). Indeed, dropout after the last LSTMs forces the classification to rely on more units. Conversely, a given LSTM unit will contribute to the prediction of more labels. Thus, the classification will be more robust, because it relies on more evidence, and might keep competing character hypotheses in a closer range, leaving more room for error correction in the beam search. Moreover, the weight halving at decoding time stabilizes the response of the network to small changes to the LSTM outputs. On the other hand, dropout on the inputs of the LSTM layer prevents it to rely on their correlations, as explained in the previous section.

TABLE IV: Effect of dropout at different combinations of places in complete Recognition Systems (RNN+LM). CERs inside parentheses correspond to Character Error Rates of the RNN alone.

	Place of dropout	Handcrafted Features		Pixels	
		WER	CER (RNN)	WER	CER (RNN)
Rimes	after all	12.8	**3.6** (6.8)	13.3	4.1 (6.5)
	before all	13.1	3.7 (**5.0**)	13.8	4.0 (5.8)
	bef. 1 / aft. 2-3	12.8	**3.6** (5.5)	13.5	4.0 (6.3)
	bef. 1-2 / aft. 3	**12.7**	**3.6** (5.6)	13.7	4.2 (6.0)
	before+after all	**12.7**	3.7 (5.4)	**12.7**	**3.9** (**5.3**)
IAM	after all	11.8	4.1 (9.0)	13.2	4.7 (10.1)
	before all	12.3	4.2 (7.9)	12.4	4.5 (8.6)
	bef. 1 / aft. 2-3	11.6	4.0 (8.2)	11.9	**4.1** (8.0)
	bef. 1-2 / aft. 3	**11.2**	**3.8** (8.1)	11.8	4.2 (8.3)
	before+after all	12.2	4.1 (**7.8**)	**11.6**	**4.1** (**7.9**)
Benth.	after all	17.3	6.9 (10.0)	19.1	7.7 (10.8)
	before all	16.6	6.2 (7.4)	17.8	6.9 (8.0)
	bef. 1 / aft. 2-3	16.1	**5.8** (**7.1**)	17.6	**6.7** (8.4)
	bef. 1-2 / aft. 3	**16.0**	6.0 (7.4)	18.1	**6.7** (8.7)
	before+after all	17.1	6.3 (7.3)	**17.5**	**6.7** (**7.5**)

In Table IV, we report the results when the different choices of dropout positions (before or after) are combined. More specifically, we apply dropout before LSTMs in lower layers, and after LSTMs in upper layers. We see that for the RNN alone (inside parentheses), dropout before some LSTMs is always better than dropout after all LSTMs, but not necessarily much better than dropout before all LSTMs and none after. On the other hand, for complete systems, adding dropout after the top LSTMs nearly always improves the results over dropout *before* only, and the best combination of *before* and *after* always outperforms both dropout before all and after all. This optimal combination seems to be before the first two and after the last LSTM for features, and before and after all LSTMs for pixels.

The final results of the complete systems using handcrafted features on evaluation sets are reported on Tables V (Rimes) and VI (IAM), with dropout before or after every LSTM, or with the best combination of both from the results of Table IV.

TABLE V: Final results on Rimes database

Inputs	Dropout	WER	CER
Features	No dropout	12.5	3.7
	After LSTM	12.7	3.8
	Before LSTM	13.2	3.8
	Before 1-2 / After 3	**12.3**	3.8
	Pham et al. [4]	**12.3**	**3.3**
	Doetsch et al. [20]	12.9	4.3

TABLE VI: Final results on IAM database. Note that [20], [21] use an open-vocabulary language model which enables them to have potentially no out-of-vocabulary words.

Inputs	Dropout	WER	CER
Features	No dropout	15.0	5.9
	After LSTM	14.5	5.7
	Before LSTM	14.0	5.3
	Before 1-2 / After 3	13.2	5.0
	Doetsch et al. [20]	12.2	4.7
	Kozielski et al. [21]	13.3	5.1
	Pham et al. [4]	13.6	5.1

V. Conclusion

In this paper, we have studied the influence of applying dropout at different positions in BLSTM-RNNs. We have shown that big differences may be observed from different relative position to the recurrent connections. Applying it only after LSTM layers, as suggested in [4], or between every layer, as suggested in [7], is not always optimal. For the RNN alone, before recurrent connections seems to be a good choice. Yet we observed, especially for complete recognition systems, that applying dropout on the classification features, *i.e.* after the last LSTM, is crucial to observe WER improvements, without much degradation of the performance of the RNN alone. It seems like dropout is best close to the inputs and outputs of the network. In future works, we plan to verify these observations with different RNN architectures, in particular the MDLSTM-RNNs.

Acknowledgment

This work was partially funded by the French Grand Emprunt-Investissements d'Avenir program through the PACTE project.

References

[1] 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.

[2] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning*, 2010.

[3] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2814–2822. [Online]. Available: http://papers.nips.cc/paper/4878-understanding-dropout.pdf

[4] 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.

[5] 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.

[6] 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.

[7] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329

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

[9] 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.

[10] 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*, 2014.

[11] 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.

[12] 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.

[13] J. A. Sánchez, V. Romero, A. Toselli, and E. Vidal, "ICFHR 2014 HTRtS: Handwritten Text Recognition on tranScriptorium Datasets," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[14] R. Buse, Z. Q. Liu, and T. Caelli, "A structural and relational approach to handwritten word recognition." *IEEE Transactions on Systems, Man and Cybernetics*, vol. 27, no. 5, pp. 847–61, Jan. 1997.

[15] A.-L. Bianne, F. Menasri, R. Al-Hajj, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem, "Dynamic and Contextual Information in HMM modeling for Handwriting Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 10, pp. 2066 – 2080, 2011.

[16] 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.

[17] D. Povey, A. Ghoshal, N. Goel, M. Hannemann, Y. Qian, P. Schwarz, J. Silovsk, and P. Motl, "The Kaldi Speech Recognition Toolkit," in *Workshop on Automatic Speech Recognition and Understandin*, 2011.

[18] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," in *Springer Handbook on Speech Processing and Speech Communication*, 2008, pp. 1–31.

[19] T. Bluche, H. Ney, and C. Kermorvant, "The LIMSI Handwriting Recognition System for the HTRtS contest," in *13th International Conference on Document Analysis and Recognition (ICDAR)*, 2015 – Submitted.

[20] P. Doetsch, M. Kozielski, and H. Ney, "Fast and robust training of recurrent neural networks for offline handwriting recognition," in *14th International Conference on Frontiers in Handwriting Recognition (ICFHR2014)*, 2014, pp. –.

[21] 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.