Bidirectional decoder networks for attention-based end-to-end offline handwriting recognition

Patrick Doetsch, Albert Zeyer, and Hermann Ney
Human Language Technology and Pattern Recognition, Computer Science Department
RWTH Aachen University
52056 Aachen, Germany
Email: $\{$*doetsch,zeyer,ney*$\}$*@cs.rwth-aachen.de*

***Abstract*—Recurrent neural networks that can be trained end-to-end on sequence learning tasks provide promising benefits over traditional recognition systems. In this paper, we demonstrate the application of an attention-based long short-term memory decoder network for offline handwriting recognition and analyze the segmentation, classification and decoding errors produced by the model. We further extend the decoding network by a bidirectional topology together with an integrated length estimation procedure and show that it is superior to unidirectional decoder networks. Results are presented for the word and text line recognition tasks of the RIMES handwriting recognition database. The software used in the experiments is freely available for academic research purposes.**

***Keywords*-long short-term memory, recurrent neural network, end-to-end, soft-attention, handwriting recognition**

I. Introduction

Sequence processing with recurrent neural networks (RNNs) and particular long short-term memories (LSTMs) [1] showed superior performance compared to previous approaches on a variety of tasks, including statistical machine translation (SMT) [2][3], as well as speech and handwriting recognition [4][5][6][7]. The goal in these tasks is to learn a program that maps a sequence of input observations x_1^T to a sequence of output symbols y_1^N. While neural networks proved as powerful tool to estimate the best output symbol given a slice of x_1^T, the segmentation and length estimation is still done using a state model. In the hybrid hidden Markov model (HMMs) approach, an HMM is trained and used to obtain a Viterbi alignment, which maps each input observation x_t to one of the symbols in the output sequence. Here, length modeling and segmentation are done by the HMM and therefore implicitly constrained by the state transitions and lengths of the symbol HMMs. While connectionist temporal classification (CTC) [8] also allows networks to do realignments after every update in order to improve the segmentation, the underlying model is still constrained by the single state / blank topology that defines the CTC layer. Consequently, neither the hybrid HMM approach nor CTC is not able to decode x_1^T to y_1^N if $N > T$. However, many successes with convolutional operations and subsampling suggest that it might be beneficial to produce less observations than output symbols and to decompress the output sequence using the total set of generated features [9].

Recently a sequence modeling technique was introduced which is able to overcome these limitations [10][11]. In these models, the processing of the input sequence is performed by a different network than the generation of the output sequence. This encoder network can hereby be any differentiable feature generator, and neural networks are a natural choice. The output sequence is generated using a recurrent decoder network, which emits the output sequence y_1^N label by label while utilizing the encoded features. A differentiable interconnection between the encoder and decoder network makes both models trainable end-to-end. This allows the model to generate an arbitrary number of discriminative features in the encoder network. The decoder network directly models the output label context just as in a neural language model, while being able to use the full set of encoded features in each step. Length modeling is performed as local classification task of the decoder network. A special end-of-sequence symbol (*eos*) is added to the label set which indicates that decoding should be stopped. Unfortunately, this only allows for unidirectional decoding, since different decoding directions might result in different output length hypotheses. It is however well known that bidirectional networks [12], and particular bidirectional LSTMs (BLSTMs), show superior performance in handwriting recognition tasks. We therefore propose in integrated length estimation component which is then used to decode the sequence from both directions.

This paper is organized as follows: In Section II we will describe the existing approaches for end-to-end attention based models in ASR and SMT. Section III will then give an overview over the system components and explain the encoder topology chosen in our experiments. Afterwards we will describe the decoder network in Section IV, including the attention mechanism and several implementation details that improved convergence, followed by our approach to estimate the output sequence length for bidirectional decoder networks in Section V. In Section VI we will evaluate our system on the RIMES handwriting recognition dataset.

II. State-of-the-art

Capturing long range contextual information in RNNs is hard due to the noise resulting from the shared representation of the memory state and the network input [13]. The LSTM architecture [14] [1] was designed to overcome large input lags by introducing gating units that are able to make a selective use of either the memory state or the network input. In the LSTM cell design an input, output and forget gate emit sigmodial activations that are multiplied with the cell input, the previous memory state or the cell output respectively. Several other cell designs have been proposed [15], but in this work we will always use the LSTM cell design.

Generative RNNs, like the decoder network, are of large interest recently and can be used in any imaginable task with natural language as output, like e.g. including image captioning [16]. Text generation with recurrent neural networks was proposed in [17] and further studied by many following publications of the authors. In [18], the decoder network was formally introduced as transducer network. An application of this framework was then published in [19], where an RNN was trained to produce synthetic handwritten text images. The breakthrough in sequence learning tasks however came with the initial results in [11], where the encoder / decoder model was introduced to implement an SMT system. The authors further describe the attention mechanism, which extends on the idea proposed in [19] to compute a linear combination of all given inputs. This topology was then quickly applied to ASR [20], showing promising results on a large private dataset. In [21] the attention model was then further refined and adapted for long sequences in monotone alignment tasks like speech and handwriting recognition. In [9] an attention-based model was applied on images of printed text in natural scenes and several encoder architectures were studied.

III. System overview

Given an input sequence of observations x_1^T, we aim to find the most probable output symbol sequence y_1^N defined over a vocabulary V by maximizing the sequence posterior probability $p(y_1^N|x_1^T)$. During training, we assume that the dependency on previous labels is modeled by the recurrent connections and therefore define the loss as symbol-wise cross-entropy of the correct label sequence z_1^N:

$$\mathcal{L}(x_1^T, z_1^N) \quad = \quad -\sum_{n=1}^{N} \log p_n(z_n|x_1^T) \tag{1}$$

The posterior probability is modeled with a regular softmax layer, which transforms the decoder outputs $dec(x_1^T) = h_1^N$ into a posterior probability distribution over V:

$$p_n(z_n|x_1^T) = \mathrm{softmax}(h_1^V) = \exp(h_i) / \sum_i \exp(h_i) \tag{2}$$

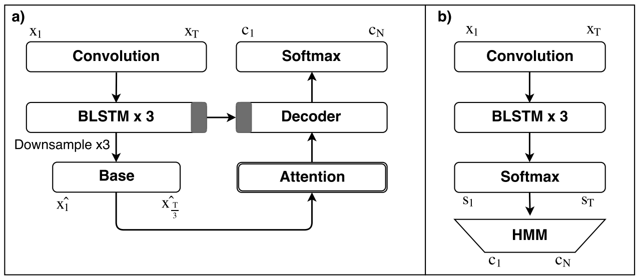

Figure 1. The encoder design. Left: An input image slice is transformed into 1344 convolutional features which are then processed by three LSTM layers followed by a downsampling step. Afterwards the resulting feature sequence is scanned with an attention mechanism to generate the output symbol sequence. Right: The same design for the framewise system, which classifies each input frame based on an initial alignment and performs the length modeling and comparative evaluation within an HMM afterwards.

The decoder outputs h_1^N are computed with an LSTM as described in the next section. The inputs x_1^T could be chosen to be the network inputs, however in practice it is useful to use an encoder network to compute discriminative representations $\hat{x}_1^R$ that allow the decoder to perform a better classification. In this work we preprocess our images similar to [6] but don't apply the PCA in the end. This results in deslanted and moment-normalized image slices of width 8 and height 32. A convolutional layer is applied to these image slices and 32 filters of size 3×3 are shifted over the image with a vertical max-pooling by 4 pixels and border overlapping, resulting in 1344 outputs. We then apply three stacked BLSTM layers with 512 cells for each direction and finally downsample the frames along the time axis by a factor of 3. The encoded feature vector $\hat{x}_1^R$ therefore has 1024 dimensions and R is roughly $T/3$. The downsampling approach is similar to [21] and we can confirm that reducing the number of inputs to the decoder is beneficial for convergence. The architecture of the encoder is depicted in Figure 1.

IV. Recurrent decoder neural networks

Decoder networks are generative neural networks that generate the label activations h_1^N based on some initial state and a memory state. They emit the labels one after another, which allows to integrate all the insights from neural language modeling. This means that the decoder can be pretrained on a lot of text data before taking any actual observation into account. In each decoding step, the network has access to the output of the encoder. One way to make use of the encoded inputs is to combine them into a single representation which is then used to initialize the decoder [10]. This requires to encode all input features into a fixed size representation, and often an LSTM is chosen to compute a final state vector v_{enc} after reading the encoded input features [22]. In this work the decoder network is a generative LSTM, which creates the output activation

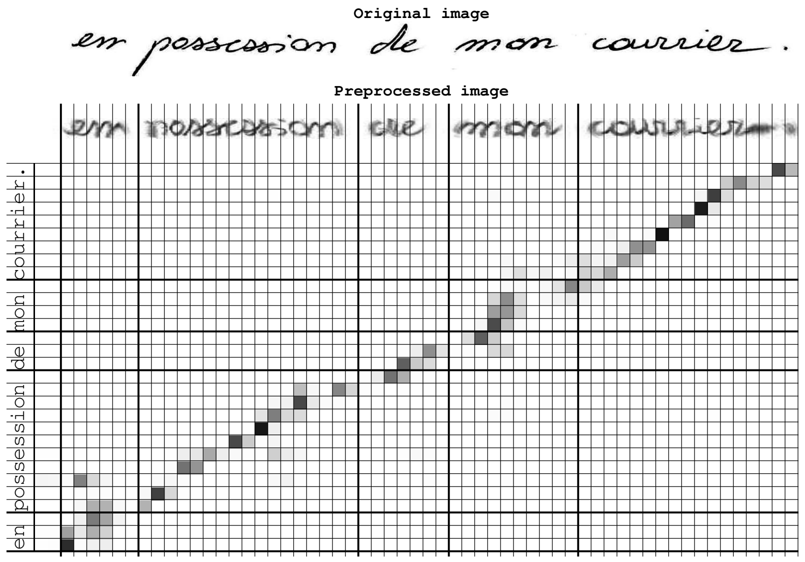

Figure 2. An example of the attention weights after reading an image containing the sentence "en possession de mon courrier." of the RIMES database. A darker color indicates a higher attention weight. At the top of the figure the source image is shown and below that a reconstruction of the moment-normalized features which are used in the attention mechanism. Word boundaries were manually emphasized with bold lines.

sequence by computing the cell state and output activation of the LSTM cell [1] based on the previous cell outputs:

$$h_n = \begin{cases} lstm(h_{n-1}) & n > 0 \\ v_{enc} & n = 0 \end{cases} \tag{3}$$

An unfavorable aspect of using only this approach is that input sequences of, in theory, arbitrary length should be encoded into a fixed size vector representation, such that a better method was proposed that enable the decoder to make use of the full set of features provided by the encoder.

A. Attention mechanism

The limited capacity of the encoder state suggests that the performance of recurrent decoder models decreases with increasing sequence lengths. The degradation of classification performance was confirmed in [11] and a mechanism was proposed to compute an expected input $\bar{x}_n$ for the decoder network based on the encoder network. This allows the decoder to reinitialize its inner state and to make local predictions without the requirement to store information for an arbitrary time span, thus allowing it to pay *attention* to a selection of inputs. In order to keep the model differentiable, this attention mechanism is implemented as a linear combination of the encoded features with normalized weights $\alpha \in \mathbb{R}^{N \times R}$:

$$\bar{x}_n = \sum_r \alpha_{nr} \hat{x}_r \tag{4}$$

where $\sum_r \alpha_{nr} = 1$ and $\alpha_{nr} \geq 0 \ \forall n, r$. The approaches to compute the weights α_1^R can roughly be classified into *content-based* methods and *location-based* methods [23]. Content-based attention mechanisms compute α_r as a normalized similarity score between the hidden state h_{n-1} and $\hat{x}_r$. The decoder network is therefore only required to create an approximate version of the encoded feature, and the similarity function serves as look-up to find the correct input. On the other hand, location-based attention mechanisms aim to compute the optimal input based on the previous attention itself. This approach is useful if a monotonic alignment can be assumed, as it is the case in handwriting recognition.

In this work we use a two level attention mechanism which makes use of both of these concepts. First, we define two matrices P and Q which project both h_{n-1} and $\hat{x}_1^R$ to a common template size, followed by a non-linear transformation σ. The weight distribution over R is obtained by calculating the euclidean distance between the projections at each encoded feature Px_r and the projected hidden state Qh_{n-1}:

$$\alpha_{nr} = \text{softmax}\left(\sqrt{\sum_i (\sigma(P\hat{x}_r) - \sigma(Qh_{n-1}))^2}\right) \tag{5}$$

In a second step we constrain α_n in a location-based fashion by taking the previous attention α_{n-1} into account as proposed in [21]. A convolutional neural network with 64 filters and a context of 2×7 pixels is trained on the weight patterns $[\alpha_{n-1}, \alpha_n] \in \mathbb{R}^{2 \times R}$. Afterwards the filters are accumulated over R and projected down into a single dimension. These filter outputs are normalized and then multiplied to the content-based estimate of α_n. This introduces a trainable transition model into the network. In [11] and [20] the method to compute α_n was further augmented by the previous softmax label, which is consistent with the view that the decoder is a neural language model. We however observed severe overfitting issues, even when regulating the amount of training label information as described in [20]. Therefore we did not include this into our model. An example of the attention weights for a model trained over 500 epochs is shown Figure 2. It can be seen that the model accurately attends on the most relevant input frames.

V. Length estimation

The decoder network outputs character by character given the decoder history and the expected input from the attention mechanism. In [20] the authors followed the SMT approach in [11] and used a unidirectional LSTM as decoder. Decoding is interrupted once a special *end-of-sequence* symbol was recognized by the network.

Here we propose a way to decode the output $y_1, \ldots, y_N$ from both directions. In order to combine the forward and backward direction over the same time axis, the required number of time steps to unroll the network has to be known. We achieve this by taking the final encoder state v_{enc} as input to a small feed-forward neural network of size k defined by a matrix $W_l \in \mathbb{R}^{|v_{enc}| \times k}$ and a vector $b_l \in \mathbb{R}^k$, that estimates the length directly using linear regression with coefficients v_{reg}:

$$\hat{N} = \langle v_{reg}, \mathrm{relu}(W_l v_{enc} + b_l) \rangle + \bar{b} \tag{6}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product and $\bar{b} \in \mathbb{R}$ is a trainable shift that we initialize with the average length in the training set. During training we extend the objective function by the squared differences of the real length and $\hat{N}$:

$$\mathcal{L}_{len}(N) = \beta(N - \hat{N})^2 \tag{7}$$

where β is a scaling factor which we set to 0.1. In order to simplify convergence, we use the correct length during training while still minimizing $\mathcal{L}_{len}$. During testing we then take the estimated length. A histogram showing the errors made by the proposed method is presented in Figure 3.

With the length estimate we are able to design the decoder in a bidirectional way. We use the same topology both for the forward and backward decoder in our experiments. The initial state of the backward decoder is initialized with the final state of the encoder LSTM in forward direction, and the initial state of the forward decoder with the final state of the encoder LSTM in backward direction. This reverse connectivity was motivated in [10] and we observed small improvements when applying it in our system.

VI. Experimental Setup

We evaluate the proposed system on two tasks of the RIMES database [24]. The RIMES database contains unconstrained French handwriting and was the dataset of

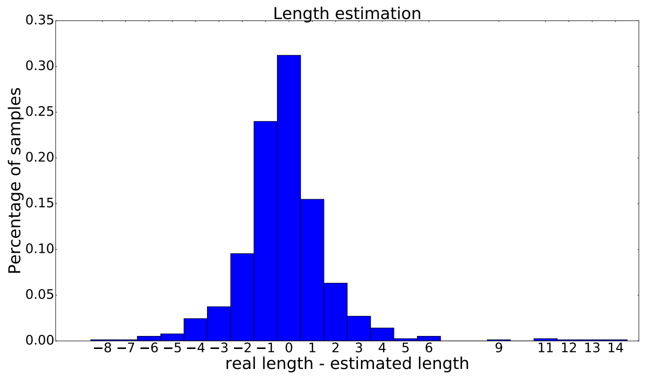

Figure 3. Length estimation error on the evaluation set of the RIMES database. Each bin corresponds to a length difference of the real sequence length and the estimated sequence length. The size of each bin shows the percentage of the samples with the corresponding difference.

the ICDAR2011 handwriting recognition competition. We use the evaluation sets of this competition as test set of our system. Two different tasks could be challenged by the participating teams. The first task was isolated word recognition with 51,738 word images in the training set and 7464 word images in the test set. A label set of 82 distinct characters was used in the output layer. The second task contained images of full text lines and was decomposed in a training set with 11,275 text line images and a test set with 778 text line images. Because of additional symbols like punctuation marks the label set on this task contains 96 distinct characters, and there are 8,537 distinct words in the vocabulary. During training of the models we take 10% of the training data as validation set and evaluate models at the point of minimal cross-entropy error and minimal frame error. Optimization was done on batches containing up to 25 sequences and 5,000 frames using the Adam [25] learning rate scaling rule with an initial learning rate of 10^{-3}.

In the experiments we are trying to understand the source of errors in the end-to-end system. We distinguish between the following types of errors: alignment errors, length estimation errors, and classification errors. Alignment errors happen if attention weights are focusing on the wrong inputs, a wrongly estimated length means that the generated sequence is too long or too short and classification errors are considered as wrong softmax activations given the correct input. In order to separate these types of errors, we consider three systems. First, we consider a traditional framewise system which uses an HMM generated alignment to perform a local classification at each input time step. We realign the system to assure that the alignment is of good quality and that most of the errors are pure classification errors. The topology of the framewise system is identical to the topology of the encoder LSTM but without the final downsampling. Secondly, we train an end-to-end system which *translates* the alignment to the correct string of characters. This can be

Table I
Length estimation error and symbol-wise label error for different input types on the RIMES database.

	words		lines	
	line error[%]	label error[%]	line error[%]	label error[%]
characters	0.0	0.0	0.3	0.9
alignment	3.5	0.3	64.6	4.7
features	18.7	3.6	75.9	13.3

seen as a SMT task which solely evaluates the alignment and length estimation problems without any classification issues. In order to do the experiment, we replaced the first layer in the encoder by a 256 dimensional linear layer in which the sparsely coded characters are embedded. Finally we trained an end-to-end system that recognizes the characters based on the text images. We used RETURNN [26], the recurrent neural network training software developed at our institute, to train all networks and an example setup to run these experiments will be provided. The systems were decoded using the RASR [27] toolkit. Experiments on framewise labeled data were decoded using a left-to-right topology with adjusted forward, skip and loop transition probabilities. Experiments with the encoder-decoder topology simply use a single state left-to-right HMM without skip or loop transition. Search was done using an adaptive beam and a unigram language model on the word recognition task and a 4-gram language model on the text-line recognition task. A dictionary that contained the test words was provided to the participants at the beginning of the evaluation phase [24] therefore was also used in our decoding. Training was performed on a NVIDIA GTX 980 GPU and took around 2 days for the encoder-decoder topology, compared to only a few hours in the hybrid HMM setting. We furthermore applied dropout in order to deal with the model complexity.

Table I shows the relative number of wrongly estimated lengths, first based on the alignment and then based on the original features. We can see that it is significantly easier for the system to compute the length based on the HMM alignment, however, even on the word based task it is not able to estimate the lengths without errors. This effect is even more observable on the line recognition task. This shows that a substantial amount of the total error occurs from memorizing and reducing the total sequence which can not result from alignment or classification errors. For comparison we also show that memorizing the character sequence itself, i.e. just copying the output character sequence, is solved almost flawlessly by the model. For the remaining experiments we always used the image data as input after applying the preprocessing described in Section III. In Table II and III we compare a unidirectional and a bidirectional decoder network using the proposed length estimation on the isolated word and text line recognition tasks. To make a fair comparison, the single unidirectional decoder had twice the capacity of the decoder networks in the bidirectional setup and was initialized with both the forward and backward encoder state. We also add the numbers of the oracle experiment where the number of output symbols N is known in advance. In total we see that length estimation errors make up a big part of the total error. However, with a bidirectional decoder network, the error rates become competitive to the framewise system. Especially on the isolated word recognition task, where the alignment problem is simpler, the end-to-end systems show basically the same performance as hybrid HMM systems. It is worth mentioning, that the optimization of end-to-end systems requires about ten times more iterations than the optimization of framewise systems. However, the important advantage of end-to-end systems is the integrated realignment process, and it can be seen in the table that realigning a framewise system also improves the final error.

Table II
Evaluation of our proposed bidirectional decoder system on the isolated word recognition task of the RIMES database together with the oracle results where the actual length is known.

Systems	WER [%]	CER [%]
Unidirectional end-to-end (known length)	5.4	2.5
Bidirectional end-to-end (known length)	3.6	1.3
Unidirectional end-to-end (estimated length)	9.5	5.4
Bidirectional end-to-end (estimated length)	7.7	4.4
Framewise	7.6	3.2
+ realignment	6.8	2.9
A2IA [24]	5.13	–
Jouve [24]	12.53	–
IRISA [24]	21.41	–
ParisTech [24]	24.88	–

Table III
Evaluation of our proposed bidirectional decoder system on the text line recognition task of the RIMES database together with the oracle results where the actual length is known. The framewise system was trained without attention and trained using a hybrid HMM approach.

Systems	WER [%]	CER [%]
Unidirectional end-to-end (known length)	16.9	8.4
Bidirectional end-to-end (known length)	14.0	7.1
Unidirectional end-to-end (estimated length)	21.6	10.8
Bidirectional end-to-end (estimated length)	16.2	8.0
Framewise	13.0	7.6
+ realignment	12.9	5.8
A2IA [28]	13.9	6.3
Telecom ParisTech [24]	26.8	15.8

VII. Conclusion

In this work we presented an attention-based recurrent neural network training framework that can be trained end-to-end. In our system, a recurrent encoder produces a set of features which are then used in the decoder in a location-based and content-based attention mechanism. We demonstrated the importance of the sequence length estimation and proposed a fast, integrated estimation method together with a bidirectional decoder network topology, which lead to consistent improvements in our experiments. Results were presented for a well known handwriting recognition task and showed that end-to-end systems are able to compete with traditional hybrid HMM systems.

References

[1] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal for Machine Learning*, vol. 3, pp. 115–143.

[2] X. Chen, X. Liu, M. Gales, and P. Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition," Brisbane, Australia, Apr. 2015, pp. 5411–5415.

[3] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 1693–1701.

[4] A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with Deep Bidirectional LSTM," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec. 2013, pp. 273–278.

[5] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," Singapore, Sep. 2014, pp. 338–342.

[6] 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 (ICFHR)*, Sept 2014, pp. 279–284.

[7] R. Messina and J. Louradour, "Segmentation-free handwritten Chinese text recognition with LSTM-RNN," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, Aug. 2015, pp. 171–175.

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*. New York, NY, USA: ACM, 2006, pp. 369–376.

[9] C. Lee and S. Osindero, "Recursive recurrent nets with attention modeling for OCR in the wild," *CoRR*, vol. abs/1603.03101, 2016.

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *arXiv:1409.3215 [cs]*, Sep. 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[12] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[13] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *CoRR*, vol. abs/1211.5063, 2012.

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

[15] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[16] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *arXiv preprint arXiv:1502.03044*, 2015.

[17] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, June 2011, pp. 1017–1024.

[18] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, 2012.

[19] ——, "Generating sequences with recurrent neural networks," *CoRR*, 2013.

[20] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," vol. abs/1508.01211, 2015.

[21] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," 2015.

[22] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," 2015. [Online]. Available: http://arxiv.org/abs/1506.06726

[23] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *CoRR*, vol. abs/1410.5401, 2014.

[24] E. Grosicki and H. El-Abed, "Icdar 2011 - french handwriting recognition competition," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, Sept 2011, pp. 1459–1463.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[26] P. Doetsch, A. Zeyer, P. Voigtlaender, I. Kulikov, R. Schlüter, and H. Ney, "RETURNN: The RWTH extensible training framework for universal recurrent neural networks," *submitted to Interspeech 2016*, 2016.

[27] D. Rybach, S. Hahn, P. Lehnen, D. Nolden, M. Sundermeyer, Z. Tüske, S. Wiesler, R. Schlüter, and H. Ney, "Rasr - the rwth aachen university open source speech recognition toolkit," in *IEEE Automatic Speech Recognition and Understanding Workshop*, Waikoloa, HI, USA, Dec. 2011.

[28] F. Menasri, J. Louradour, A.-L. Bianne-Bernard, and C. Kermorvant, "The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition." in *DRR*, ser. SPIE Proceedings, C. Viard-Gaudin and R. Zanibbi, Eds., vol. 8297. SPIE, 2012.