END-TO-END ATTENTION-BASED LARGE VOCABULARY SPEECH RECOGNITION

Dzmitry Bahdanau[†], *Jan Chorowski*[‡], *Dmitriy Serdyuk*[†], *Philémon Brakel*[†] *and Yoshua Bengio*[†1]

[†] Université de Montréal
[‡]University of Wrocław
[1] CIFAR Fellow

ABSTRACT

Many state-of-the-art Large Vocabulary Continuous Speech Recognition (LVCSR) Systems are hybrids of neural networks and Hidden Markov Models (HMMs). Recently, more direct end-to-end methods have been investigated, in which neural architectures were trained to model sequences of characters [1, 2]. To our knowledge, all these approaches relied on Connectionist Temporal Classification [3] modules. We investigate an alternative method for sequence modelling based on an attention mechanism that allows a Recurrent Neural Network (RNN) to learn alignments between sequences of input frames and output labels. We show how this setup can be applied to LVCSR by integrating the decoding RNN with an n-gram language model and by speeding up its operation by constraining selections made by the attention mechanism and by reducing the source sequence lengths by pooling information over time. Recognition accuracies similar to other HMM-free RNN-based approaches are reported for the Wall Street Journal corpus.

Index Terms— neural networks, LVCSR, attention, speech recognition, ASR

1. INTRODUCTION

Deep neural networks have become a popular replacement for Gaussian Mixture Models for acoustic modeling in state-of-the-art large vocabulary speech recognition systems [4]. However, these networks are typically still trained to match acoustic frames to the states of classical systems in hybrid setups that rely on multi-stage training procedures. The complexity of the resulting systems may explain why research about end-to-end trainable neural HMM-free systems for LVCSR is becoming popular. Recurrent Neural Network models trained with Connectionist Temporal Classification (CTC) [3] achieved promising results on the Wall Street Journal [5, 1] and on the Switchboard [2] corpora. Both of these models were trained to predict sequences of characters and were later combined with a word level language model.

While the results of CTC based systems are promising, alternative methods for neural sequence modelling have been recently proposed. For instance, a new direction of neural network research has emerged that deals with models that learn to focus their "attention" to specific parts of their input. Systems of this type have been successfully applied to a variety of tasks including machine translation [6], caption generation [7], handwriting synthesis [8], visual object classification [9] and phoneme recognition [10, 11].

In this work, we investigate the application of an Attention-based Recurrent Sequence Generator (ARSG) as a part of an end-to-end LVCSR system. We start from the system proposed in [11] for phoneme recognition and make the following contributions:

1. We reduce total training complexity from quadratic to linear, by applying the concept of windowing [11] during both network training and application.
2. In the spirit of he Clockwork RNN [12] and the hierarchical gating RNN [13], we introduce a recurrent architecture that successively reduces the source sequence length by pooling frames neighboring in time. [1]
3. We show how a character-level ARSG and $n-$gram word-level language model can be combined into a complete system using the Weighted Finite State Transducers (WFST) framework.

2. ATTENTION-BASED RECURRENT SEQUENCE GENERATORS FOR SPEECH

The proposed system is an encoder-decoder [15, 16] network that can map sequences of speech frames to sequences of characters[2]. It consists of a deep bi-directional RNN that *encodes* the speech signal into a suitable feature representation and of an Attention-based Recurrent Sequence Generator that *decodes* this representation into a sequence of characters.

2.1. Encoder-Decoder Architecture

We use a deep Bidirectional RNN (BiRNN) as the encoder [17]. Thus, the representation is a sequence of BiRNN state

[1]This mechanism has been recently independently proposed in [14].

[2]The word "decoder" refers to a network in this context, not to the final recognition algorithm.

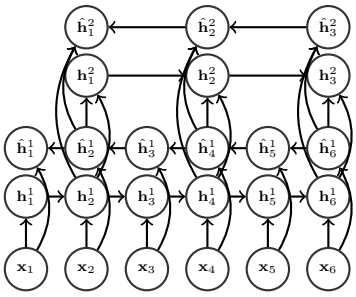

Fig. 1. A pooling over time BiRNN: the upper layer runs twice as slow as the lower one. It can average, or subsample (as shown in the figure) hidden states of the layer below it.

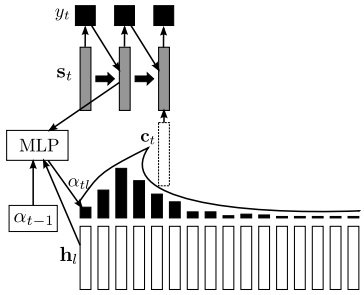

Fig. 2. Schematic representation of the Attention-based Recurrent Sequence Generator. At each time step t, an MLP combines the hidden state $\mathbf{s}_{t-1}$ with all input vectors $\mathbf{h}_l$ to compute the attention weights α_{tl}. Then, the new hidden state $\mathbf{s}_t$ and prediction for output label y_t can be computed.

vectors $(\mathbf{h}_1, \ldots, \mathbf{h}_N)$. For a standard deep BiRNN, the sequence $(\mathbf{h}_1, \ldots, \mathbf{h}_N)$ is as long as the input of the bottom-most layer, which in the context of speech recognition means one $\mathbf{h}_i$ for every 10ms of the recordings. We found that for our decoder (see 2.2) such a representation is overly precise and contains much redundant information. This led us to add pooling between BiRNN layers as illustrated by Figure 1.

2.2. Attention-equipped RNNs

The decoder network in our system is an Attention-based Recurrent Sequence Generator (ARSG) [6, 11]. An ARSG produces an output sequence $(y_1, \cdots, y_T)$ one element at a time, simultaneously aligning each generated element to the encoded input sequence $(\mathbf{h}_1, \cdots, \mathbf{h}_N)$. It is composed of an RNN and an additional subnetwork called the "attention mechanism". The attention mechanism selects the temporal locations over the input sequence that should be used to update the hidden state of the RNN and to predict the next output value. The selected input sequence elements are combined in a weighted sum $\mathbf{c}_t = \sum_n \alpha_{tn}\mathbf{h}_n$, where α_{tn} are called the attention weights and we require that $\alpha_{tn} \geq 0$ and that $\sum_n \alpha_{tn} = 1$. We picture and ARSG in Figure 2.

The equations describing the attention mechanism are [11]:

$$\mathbf{F} = \mathbf{Q} * \boldsymbol{\alpha}_{t-1} \tag{1}$$

$$e_{tn} = \mathbf{w}^\top \tanh(\mathbf{W}\mathbf{s}_{t-1} + \mathbf{V}\mathbf{h}_n + \mathbf{U}\mathbf{f}_n + \mathbf{b}) \tag{2}$$

$$\alpha_{tn} = \exp(e_{tn}) \Big/ \sum_{n=1}^{N} \exp(e_{tn}) \; . \tag{3}$$

where $\mathbf{W}$, $\mathbf{V}$, $\mathbf{U}$, $\mathbf{Q}$ are parameter matrices, $\mathbf{w}$ and $\mathbf{b}$ are parameter vectors, $*$ denotes convolution, $\mathbf{s}_{t-1}$ stands for the previous state of the RNN component of the ARSG.

Simply put, the attention mechanism combines information from three sources to decide where to focus at the step t: the decoding history contained in $\mathbf{s}_{t-1}$, the content in the candidate location $\mathbf{h}_n$ and the focus from the previous step described by the attention weights $\boldsymbol{\alpha_{t-1}}$. It is shown in [11] that using $\boldsymbol{\alpha}_{t-1}$ in the equations defining $\boldsymbol{\alpha}_t$ is crucial for reliable behaviour on long input sequences.

A disadvantage of the approach from [11] is the complexity of the training procedure, which is $O(TN)$ since weights α_{tn} have to be computed for all pairs of input and output positions. The same paper showcases a windowing approach that reduces the complexity of decoding to $O(T + N)$. In this work, we apply the windowing at the training stage as well. Namely, to predict the character y_t we constrain the attention mechanism to consider positions from the range $(m_{t-1} - w_{left}, \ldots, m_{t-1} + w_{right})$, where m_{t-1} is the median of $\boldsymbol{\alpha}_{t-1}$, interpreted in this context as a distribution. The values w_{left} and w_{right} define how much the window extends to the left and to the right respectively. This modification makes training significantly faster.

Apart from the speedup it brings, windowing can be also very helpful for starting the training procedure. From our experience, it becomes increasingly harder to train an ARSG completely from scratch on longer input sequences. We found beneficial to provide a rough estimate of the desired alignment during the early training stage. Specifically, we forced the network to choose from positions in the range $R_t = (s_{min} + tv_{min}, \ldots, s_{max} + tv_{max})$. The numbers s_{min}, s_{max}, v_{min}, v_{max} were roughly estimated from the training set so that the number of leading silent frames for training utterances was between s_{min} and s_{max} and so that the speaker speed, i.e., the ratio between the transcript and the encoded input lengths, was between v_{min} and v_{max}. We aimed to make the windows R_t as narrow as possible, while keeping the invariant that the character y_t was pronounced within the window R_t. The resulting sequence of windows is quickly expanding as the character number t grows, but still it helped to quickly move the network out of the random initial mode, in which it had often aligned all characters to a single location in the audio data. The median-centered windowing could not be used for this purpose, since it relies on the quality of the previous alignment to define the window for the new one.

3. INTEGRATION WITH A LANGUAGE MODEL

Although an ARSG implicitly learns how an output symbol depends on the previous ones, the transcriptions of the training utterances are typically insufficient to learn a high-quality language model. For this reason, we investigate how a character-level ARSG can be integrated with a word-level language model.

We use the Weighted Finite State Transducer (WFST) framework [18, 19] to build a character-level language model from a word-level one. We compose the language model Finite State Transducer (FST) G with a lexicon FST L that spells out the letters of each word. For moderately sized language models we build an FST $T = \min(\det(L \circ G))$ to define the log-probability for character sequences. We push the weights of this FST towards the starting state to help hypothesis pruning during decoding. For larger (e.g. trigram) language models the determinization is impractically slow. To handle non-deterministic FSTs, we assume that its weights are in the logarithmic semiring and compute the total log-probability of all FST paths corresponding to a character prefix from the beam.

The decoding algorithm uses a left-to-right beam search [16] to find the transcript y that minimizes the cost $\mathcal{L}$ which combines the encoder-decoder (ED) and the language model (LM) outputs as follows:

$$\mathcal{L} = -\log p_{ED}(y|x) - \beta \log p_{LM}(y) - \gamma T \qquad (4)$$

where β and γ are tunable parameters, T is the length of the output sequence $(y_1, \ldots, y_T)$. The last term γT is important, because without it the LM component dominates and the cost $\mathcal{L}$ is minimized by too short sequences. We note that the same criterion for decoding was proposed in [1] for a CTC network.

4. RELATION TO PRIOR WORK

A popular method to train sequence prediction networks is Connectionist Temporal Classification [3]. It has been successfully used for phoneme recognition [17] and character-based LVCSR [5, 1, 2, 20]. CTC allows recurrent neural networks to predict sequences that are shorter than the input sequence by summing over all possible alignments between the output sequence and the input of the CTC module. In the CTC approach, output labels are conditionally independent given the alignment and the input sequence. In the context of speech recognition, this means that a CTC network lacks a language model, which greatly boosts the system performance when added to a trained CTC network [1, 20].

An extension of CTC is the RNN Transducer which combines two RNNs into a sequence transduction system [21, 22]. Unlike CTC, RNN transduction systems can also generate output sequences that are longer than the input. RNN Transducers have led to state-of-the-art results in phoneme recognition on the TIMIT dataset [17], which were recently matched by an ASRG network [11].

The RNN Transducer and ARSG approaches are roughly equivalent in their capabilities. Both learn an implicit language model jointly with the rest of the network. However, their main difference is that in ARSG the alignment is explicitly computed by the network, as opposed to dealing with a distribution of alignments in the RNN Transducer.

Finally, two recently published works partially overlap with the content of this paper. [14] proposes a similar to ours, character-based Encoder-Decoder network that employs pooling between BiRNN layers. [20] advocates using FSTs to build a character-level model from an n-gram model. We note, that the research described in this paper was carried independently and without communication with the authors of both aforementioned works.

5. EXPERIMENTS

5.1. Data

We trained and evaluated our models[3] on the Wall Street Journal (WSJ) corpus (available as LDC93S6B and LDC94S13B). Training was done on the 81 hour long SI-284 set of about 37K sentences. As input features, we used 40 mel-scale filterbank coefficients together with the energy and their deltas and delta-deltas, yielding a total of 123 feature values per frame. Evaluation was done on the "eval92" evaluation set. Hyperparameter selection was performed on the "dev93" set. For language model integration, we used the 20K closed vocabulary setup and the bigram and trigram language models that were provided with the data set. In addition we tried an extended language model as produced by Kaldi s5 recipe. We use the same text preprocessing as in [1], leaving only 32 distinct labels: 26 letters, apostrophe, period, dash, space, noise and end-of-sequence tokens.

5.2. Training

Our model used 4 layers of 250 forward + 250 backward GRU [15] units in the encoder, with the top two layers reading every second of hidden states of the network below it (see Figure 1), yielding a fourfold source sequence length reduction. A centered convolution filter of width 201 was used in the attention mechanism to extract a single feature from the previous step alignment as described in (3).

We used a rough estimate of the proper alignment for the first training epoch as described in Section 2.2. After that, the training was restarted with the windowing described in the same section. The window parameters were $w_{left} = w_{right} = 100$, which corresponds to considering a large 8 second long span of audio data at each step, taking into account the pooling done between layers.

[3] Our code is available at https://github.com/rizar/attention-lvcsr

Table 1. Character Error Rate (CER) and Word Error Rate (WER) on WSJ. Note that phone-based and character-based systems are not directly comparable.

Model	CER	WER
Encoder-Decoder	6.4	18.6
Encoder-Decoder + bigram LM	5.3	11.7
Encoder-Decoder + trigram LM	4.8	10.8
Encoder-Decoder + extended trigram LM	3.9	9.3
Graves and Jaitly (2014) [5],		
CTC	9.2	30.1
CTC, expected transcription loss	8.4	27.3
Hannun et al. (2014) [1],		
CTC	10.0	35.8
CTC + bigram LM	5.7	14.1
Miao et al. (2015) [20],		
CTC for phonemes + lexicon	-	26.9
CTC for phonemes + trigram LM	-	7.9
CTC + trigram LM	-	9.0
CTC + extended trigram LM	-	7.3

All weights were initialized from the normal distribution with variance 0.1. We used the AdaDelta [23] training algorithm with hyperparameters $\rho = 0.95$ and ϵ annealed in steps from 10^{-8} to 10^{-10}. The column norm constraint 1 was imposed on all weight matrices [24].

5.3. Decoding and Evaluation

For best performance we used a beam size of 100. However, this brought only $\approx 10\%$ relative improvement over a beam size of 10. We used $\alpha = 0.5$ and $\gamma = 1$ with a language model and $\gamma = 0.1$ without one. It was necessary to use an asymmetric window for the attention when decoding with large γ. More specifically, we reduced w_{left} to 10. Without this trick, the cost $\mathcal{L}$ could be infinitely minimized by looping across the input utterance, for the penalty for jumping back in time included in $\log p(y|x)$ was not high enough.

5.4. Results

Results of our experiments are gathered in Table 1. Our model shows performance superior to that of CTC systems when no external language model is used. The improvement from adding an external language model is however much larger for CTC-based systems. The final performance of our model is better than the one reported in [1] (12.7% vs 14.1%), but worse than the the one from [20] (11.5% vs 9.0%) when the same language models are used.

6. CONCLUSIONS

In this work we showed how an Encoder-Decoder network with an attention mechanism can be used to build an LVCSR system. The resulting approach is significantly simpler than the dominating HMM-DNN one, with fewer training stages, much less auxiliary data and less domain expertise involved. Combined with a trigram language model our system shows decent, although not yet state-of-the-art performance.

We present two methods to improve the computational complexity of the investigated model. First, we propose pooling over time between BiRNN layers to reduce the length of the encoded input sequence. Second, we propose to use windowing during training to ensure that the decoder network performs a constant number of operations for each output character. Together, these two methods facilitate the application of attention-based models to LVCSR.

Unlike CTC networks, our model has an intrinsic language-modeling capability. We believe it to be the reason for greater performance of the ARSG model when no external language model is given. However, the small size of the training corpus makes it easy to overfit on it. This may explain why the benefits of adding an external language model are smaller than for CTC networks. On the other hand, it is possible to use the states of a pre-trained language model as additional inputs to an ARSG, possibly reducing the incentive to memorize the training prompts. Joint training of the ARSG and an external language model is also possible [25]. Investigations in this direction are likely to be a part of our future work.

Acknowledgments

The experiments were conducted using Theano [26, 27], Blocks and Fuel [28] libraries.

The authors would like to acknowledge the support of the following agencies for research funding and computing support: National Science Center (Poland) grant 2014/15/D/ST6/04402, NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. Bahdanau also thanks Planet Intelligent Systems GmbH and Yandex.

7. REFERENCES

[1] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns," *arXiv preprint arXiv:1408.2873*, 2014.

[2] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al., "Deepspeech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[3] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *ICML-06*, 2006.

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[5] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *ICML-14*, 2014.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations*, 2015.

[7] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML-15*, 2015.

[8] A. Graves, "Generating sequences with recurrent neural networks," *arXiv:1308.0850*, Aug. 2013.

[9] V. Mnih, N. Heess, A. Graves, et al., "Recurrent models of visual attention," in *NIPS*, 2014, pp. 2204–2212.

[10] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent NN: First results," *arXiv:1412.1602 [cs, stat]*, Dec. 2014.

[11] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015, to appear.

[12] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork RNN," in *ICML-14*, 2014.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *ICML-15*, 2015.

[14] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv:1508.01211 [cs, stat]*, Aug. 2015.

[15] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Empirical Methods of Natural Language Processing*, Oct. 2014.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.

[17] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*. IEEE, 2013, pp. 6645–6649.

[18] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, Jan. 2002.

[19] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Implementation and Application of Automata*, J. Holub and J. Žďárek, Eds., number 4783 in Lecture Notes in Computer Science, pp. 11–23. Springer Berlin Heidelberg, Jan. 2007.

[20] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," *arXiv:1507.08240 [cs]*, July 2015.

[21] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[22] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "High-dimensional sequence transduction," in *ICASSP*. IEEE, 2013, pp. 3178–3182.

[23] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[24] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[25] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.

[26] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[27] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS Workshop, 2012.

[28] B. van Merriënboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, and Y. Bengio, "Blocks and fuel: Frameworks for deep learning," *arXiv:1506.00619 [cs, stat]*, June 2015.