INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Direct Acoustics-to-Word Models for English Conversational Speech Recognition

Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon, Michael Picheny, David Nahamoo

IBM T. J. Watson Research Center, Yorktown Heights, NY
{kaudhkha,bhuvana,saon,picheny,nahamoo}@us.ibm.com

Abstract

Recent work on end-to-end automatic speech recognition (ASR) has shown that the connectionist temporal classification (CTC) loss can be used to convert acoustics to phone or character sequences. Such systems are used with a dictionary and separately-trained Language Model (LM) to produce word sequences. However, they are not truly end-to-end in the sense of mapping acoustics directly to words without an intermediate phone representation. In this paper, we present the first results employing direct acoustics-to-word CTC models on two well-known public benchmark tasks: Switchboard and CallHome. These models do not require an LM or even a decoder at run-time and hence recognize speech with minimal complexity. However, due to the large number of word output units, CTC word models require orders of magnitude more data to train reliably compared to traditional systems. We present some techniques to mitigate this issue. Our CTC word model achieves a word error rate of 13.0%/18.8% on the Hub5-2000 Switchboard/CallHome test sets without any LM or decoder compared with 9.6%/16.0% for phone-based CTC with a 4-gram LM. We also present rescoring results on CTC word model lattices to quantify the performance benefits of a LM, and contrast the performance of word and phone CTC models.

Index Terms: automatic speech recognition, neural networks, end-to-end.

1. Introduction

Feed-forward, recurrent, and convolutional deep neural networks (DNNs) have significantly improved the state-of-the-art in acoustic models (AMs) [1–3]. Advanced neural network language models (LMs) [4, 5] and exponential LMs [6] have also significantly out-performed count-based N-gram LMs. Despite these advances, building a state-of-the-art automatic speech recognition (ASR) system is still a cumbersome multi-step exercise. This is fundamentally linked to the mathematical framework under which current ASR systems operate. ASR is typically formulated as the following maximum a-posteriori (MAP) optimization problem of finding the best word sequence given the acoustics [7]:

$$\mathbf{w}_{\text{MAP}} = \arg\max_{\mathbf{w}} P_{\text{ASR}}(\mathbf{w}|\mathbf{A};\Theta_{\text{ASR}}) \ . \quad (1)$$

Here $\mathbf{A}$ is a $T \times D$ matrix of D-dimensional acoustic feature vectors over T time steps and $\mathbf{w}$ is the sequence of words. Bayes' theorem splits this model into two:

$$P_{\text{ASR}}(\mathbf{w}|\mathbf{A};\Theta_{\text{ASR}}) \propto P_{\text{AM}}(\mathbf{A}|\mathbf{w};\Theta_{\text{AM}})P_{\text{LM}}(\mathbf{w};\Theta_{\text{LM}}) \ . \quad (2)$$

Here $P_{\text{AM}}(\mathbf{A}|\mathbf{w};\Theta_{\text{AM}})$ gives the AM probability of the acoustics $\mathbf{A}$ given the words $\mathbf{w}$, and $P_{\text{LM}}(\mathbf{w};\Theta_{\text{LM}})$ gives the LM probability of the word sequence $\mathbf{w}$. The ASR training process thus splits into two disjoint learning problems, one each for the AM and the LM. It makes decoding a convoluted process of the fusion of AM and LM log-likelihoods for many candidate word sequences before picking the most likely word sequence. Training the popular "hybrid" DNN-hidden Markov Model (HMM) AMs also involves multiple steps.

Recent research has led to "end-to-end" learning which aims to simplify the above process. This includes the connectionist temporal classification (CTC) loss [8] and the encoder-decoder framework [9, 10]. In particular, the CTC loss enables training AMs by mapping acoustics to phone or character sequences without any frame-level alignment [11–17]. However, phone/character-based CTC AMs are not truly end-to-end acoustics-to-word since they require a dictionary and an externally-trained LM during decoding to perform well.

Direct acoustics-to-word CTC models are a natural step towards true end-to-end ASR. The work by Sak et al. [16], and the more recent work by Soltau, Liao, and Sak [18] has presented direct acoustics-to-word CTC models. The latter trained word CTC models on around 125,000 hours of speech from a YouTube data set and a vocabulary of 100,000 words, and achieved results close to the state-of-the-art. This raises many interesting research questions: How well do word CTC models perform on standard public benchmarks that have advanced the state-of-the-art in ASR over the last few decades? Do they only work in a brute-force data-intensive setting? How do word CTC models scale across languages and in low-resource conditions?

To help address these questions, we present the first results using direct acoustics-to-word CTC models on two well-known public benchmark data sets - Switchboard and CallHome. CTC word models require orders of magnitude more data to train reliably compared to phone/character-based models. We present two techniques to improve the training of these models on 300 hours of Switchboard and 2000 hours of Switchboard+Fisher training sets. These include incorporating

1. Phone CTC models and hierarchical CTC [19].
2. GloVe word embeddings [20] to capture LM-like word co-occurrence information.

We find that both these techniques improve training convergence and performance of the word CTC models compared to random initialization. Our CTC word model achieves a word error rate of 13.0%/18.8% on the Hub5-2000 Switchboard/CallHome test sets without any LM or decoder compared with 9.6%/16.0% for phone-based CTC with a 4-gram LM.

The next section presents our phone CTC system, while Section 3 describes our word CTC system, various strategies we adopted to improve its performance, and an error-based comparison of the word and phone CTC systems. We conclude the paper in Section 4 with directions for future work.

The authors thank Yajie Miao and Florian Metze of Carnegie Mellon University, and Stanley Chen of IBM for helpful discussions.

http://dx.doi.org/10.21437/Interspeech.2017-546

2. Baseline Phone CTC System

Before discussing our phone CTC system, we first present a quick overview of the CTC loss function [8] for completeness.

2.1. CTC Loss

Let $\mathbf{y}$ denote the length-L target symbol sequence consisting of phones, characters, or words. Let $\mathbf{A}$ denote the $T \times D$ matrix of D-dimensional acoustic feature vectors over T time steps. The conventional cross-entropy loss requires L to be equal to T. A common approach to solve this problem is by force-aligning $\mathbf{y}$ with $\mathbf{A}$, which yields a mapping from any time $t \in \{1, \ldots, T\}$ to a symbol $y_l \in \mathbf{y}$. Instead, CTC allows an extra "blank" symbol ϕ that expands the length-L sequence $\mathbf{y}$ to a set of length-T sequences $\Omega(\mathbf{y})$, where each sequence $\tilde{\mathbf{y}} \in \Omega(\mathbf{y})$ reduces to $\mathbf{y}$ after the following two operations in sequence:

1. Removal of all repeating symbols.
2. Removal of the blank symbol ϕ.

The set $\Omega(\mathbf{y})$ is often called the set of CTC "paths" corresponding to the symbol sequence $\mathbf{y}$. The negative CTC loss is then

$$P(\mathbf{y}|\mathbf{A}) = \sum_{\tilde{\mathbf{y}} \in \Omega(\mathbf{y})} P(\tilde{\mathbf{y}}|\mathbf{A}) = \sum_{\tilde{\mathbf{y}} \in \Omega(\mathbf{y})} \prod_{t=1}^{T} P(\tilde{y}_t|\mathbf{a}_t) \; . \quad (3)$$

Dynamic programming efficiently computes the above function using the forward-backward recursion. It is easy to see that CTC implicitly constructs a left-to-right HMM with $2L+1$ states for a L-length sequence $\mathbf{y}$ by interpolating the symbol states with blank states that can be optionally skipped. A neural network predicts the probability of occupying one of the $2L+1$ HMM states at each of the T time steps. The forward-backward algorithm on this trellis computes the CTC loss (3) and its gradients, which are then back-propagated through the neural network [8].

2.2. Training and Testing Data Sets

We used 262 hours of segmented speech from the standard 300-hour Switchboard-1 audio with transcripts provided by Mississippi State University for training the 300-hour systems. We added 1698 hours of audio from the Fisher data collection and 15 hours from the CallHome audio to build the (approximately) 2000-hour systems. We built two LMs for decoding the phone CTC models and rescoring the word CTC models. The "small" 4-gram LM used 24M words from the 2000-hour audio training data with a vocabulary size of 30k words, while the "large" 4-gram LM used a vocabulary of 85k words with an additional 560M words from several public text data sets from LDC [2].

2.3. Phone CTC Model

We trained our phone CTC system over 44 phones from the Switchboard pronunciation lexicon [2] plus the blank symbol. We extracted 40-dimensional logMel filterbank energies over 25 ms frames every 10 ms from the input speech signal, stacked two successive frames, and dropped every alternate frame resulting in 80-dimensional logMel features at half the rate of the original 40-dimensional features. This "stacking+decimation" [16] provided significant speed-up in training because the sequence length reduces by half with no loss in performance. We also used 100-dimensional i-vectors for each speaker [21] and appended them to each feature, resulting in 180-dimensional feature vectors.

We used the Torch toolkit [22] with the cuDNN v5 [23] backend to build bidirectional long short-term memory (BLSTM) networks with 5 layers and 320 neurons each in the forward and backward layers. This BLSTM feeds into a fully-connected linear layer of size 640×45 followed by the softmax activation function. We used the Torch binding of the Warp-CTC tool [24] for computing the CTC loss and its gradients. We prepared the training data by sorting the utterances in decreasing order of number of frames [14], and dividing them into batches of 48 utterance each. We used stochastic gradient descent (SGD) with a learning rate of 5×10^{-4}, and cut the learning rate by half whenever the heldout loss did not decrease by more than 10%. All parameters of the neural network assumed uniformly random initial values in $(-0.01, 0.01)$. We also clipped the gradients to lie in $(-10, 10)$ to stabilize training. All our models converged in 15-20 epochs.

We constructed our CTC decoding graph similar to the one used in [14]. Table 1 shows the word error rate (WER) of our phone CTC system on the Hub5-2000 Switchboard and CallHome test sets. We see that using the big LM gives a consistent gain of around 0.5% absolute in WER. For reference, our best 6-layer 2000-hour hybrid BLSTM using FMLLR+i-vector features with 32,000 context-dependent states with standard cross-entropy+sequence training has a WER of 7.7%/14.0% on Switchboard/CallHome [2] with the big LM. However, decoding with the phone CTC model is almost 2x faster and 8x less memory-intensive than the hybrid BLSTM model.

Table 1: *This table shows the WERs of our phone CTC models on the Switchboard (SWB) and CallHome (CH) Hub5-2000 test sets for both the "small" and "big" N-gram LMs.*

Hours	*LM*	*SWB*	*CH*
300	Small	14.5	25.1
300	Big	**13.9**	**24.7**
2000	Small	10.2	16.5
2000	Big	**9.6**	**16.0**

3. Word CTC Model

Word CTC models require orders of magnitude more data to train reliably compared to conventional ASR systems. To mitigate this huge need for data, we restricted the vocabulary to contain words with at least 5 acoustic examples in the training data. This resulted in a vocabulary of approximately 10,000 words for the 300-hour system and approximately 25,000 words for the 2000-hour system. We created a special token to denote all out-of-vocabulary (OOV) words. The OOV rate for the 300-hour training data was 1.5% with the 10,000 word vocabulary, while the corresponding OOV rate for the 2000-hour training data was 0.5% for the 25,000 word vocabulary. Decoding involved a simple forward pass of the acoustic sequence through the network followed by picking the highest-scoring word at each time step and repetition/blank removal. We mapped the OOV token to the silence symbol for scoring.

Similar to the phone CTC model, we setup a 5-layer BLSTM for the word CTC model with 320 forward and backward hidden neurons in each layer. The final dense layer mapped 640 dimensions to the vocabulary size+1 (for the blank symbol). Our first training attempt used uniformly-random weight initialization over $(-0.01, 0.01)$ similar to the phone CTC model. However, the training failed to converge as shown in Figure 1, despite tuning the learning rate and the amount of

gradient clipping. We hypothesized that training the BLSTM to learn a direct mapping from acoustics to words from flat-start is a difficult proposition, especially for small data sets. We next discuss some techniques to mitigate this issue.

3.1. BLSTM Initialization with Phone CTC Model and Hierarchical CTC

As a first step, we initialized the word CTC BLSTM with the phone CTC BLSTM, with the intuition that the ability to detect sub-word units provides a good starting point for detecting words. We initialized the final dense layer randomly as before. Figure 1 shows the training and validation CTC loss using random initialization and with initialization using the phone BLSTM network. The word CTC model fails to converge with random initialization (red curves) but converges nicely when using the phone BLSTM model as a starting point (green curves).

We also experimented with hierarchical CTC [19,25] where the bottom 4 BLSTM layers were initialized with a pre-trained phone CTC model and the top BLSTM layer is randomly initialized. Training then proceeds in multi-task fashion with a weight of α to the phone CTC loss computed by branching-off from the 4th BLSTM layer and $1-\alpha$ to the final word CTC loss. After several experiments with different α, we observed that while hierarchical CTC was able to out-perform full random initialization, it was always worse compared with simply initializing the entire BLSTM network with the phone CTC BLSTM and training with only the word CTC loss.

3.2. Dense Layer Initialization with Word Embeddings

Encouraged by the impact of initializing the word BLSTM with the phone BLSTM, we wanted to explore better ways of initializing the final dense layer as well. The weights of the final layer are a set of 10,000 640-dimensional vectors, one for each word in the vocabulary. At each time step, the dense layer computes a dot product between the hidden representation and the weight vector corresponding to each word, resulting in a scalar score for each word. Words co-occurring frequently in natural text are likely to have high scores closer to each other in time. We can thus think of each 640-dimensional weight vector as a word embedding that captures word co-occurrence information.

This intuition led us to use GloVe embeddings [20] for initializing the final dense layer, similar to the technique proposed in [26] for augmenting neural network LMs. GloVe is similar in spirit to word2vec [27], but captures semantic information through a bilinear approximation of the word co-occurrence matrix $\mathbf{C}$ computed over a training text corpus. GloVe estimates the $V \times D$ matrix $\mathbf{G}$ containing D-dimensional embeddings for V words as follows:

$$\mathbf{G}^*, \mathbf{b}^* = \arg\min_{\mathbf{G},\mathbf{b}} \sum_{i,j=1}^{V} f(\mathbf{C}(i,j)) \Big(\mathbf{G}(i)\mathbf{G}(j)^T + b(i) + b(j) - \log \mathbf{C}(i,j) \Big)^2, \quad (4)$$

where $\mathbf{G}(i)$ denotes the i^{th} row of $\mathbf{G}$ and f is a weighting function. We trained two sets of 640-dimensional GloVe embeddings: one on the 24M word corpus used for training the small LM, and another on the 560M word corpus used for the big LM. We assigned random vectors for the blank symbol and OOV token. We normalized each word vector to have a unit L2 norm and scaled it by 0.1. Figure 1 shows that the GloVe initialization of the final dense layer by the average of the small and big LM embeddings yields significant improvement in training and heldout CTC loss. Table 2 shows a 3.6%/3.7% absolute improvement in WER for Switchboard/CallHome by using GloVe initialization over the phone BLSTM initialization. We also note that GloVe embeddings trained on the bigger 24M+540M word data set yield around 1% improvement in WER over the embeddings trained only on the 24M word set.

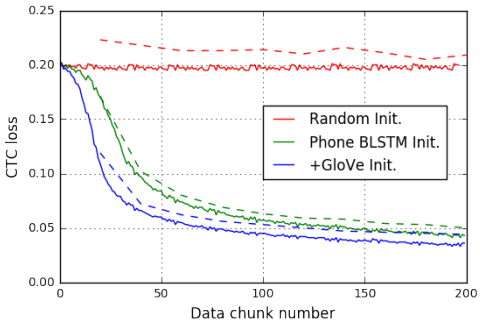

Figure 1: *This figure shows the word CTC model training (solid) and heldout (dashed) losses for the first 10 epochs over the 300-hour training set using different initialization schemes. Each data chunk on the x-axis corresponds to roughly 15 hours.*

Table 2: *This table shows the Switchboard and CallHome WERs for different initialization schemes for the 300 hour CTC word models. The randomly-initialized model failed to converge.*

Init. Method	*SWB*	*CH*
Random Init.	-	-
Phone BLSTM Init.	24.4	34.1
+ GloVe Init. (24M)	21.7	31.5
+ GloVe Init. (560M)	**20.8**	**30.4**

3.3. Final 2000-Hour Word CTC Model

We initialized the 2000-hour word CTC model with the BLSTM from the trained 2000-hour phone CTC model and GloVe embeddings trained over the 24M+560M word text corpus. We used the same input acoustic featurs as before with frame stacking+decimation. The output vocabulary was 25,000 words with a training set coverage of approximately 99.5%.

Table 3: *This table shows the Switchboard/CallHome WERs for our 2000-hour word and phone CTC models.*

AM	*LM*	*SWB*	*CH*
Word CTC	-	**13.0**	**18.8**
Word CTC	Big (rescoring)	**12.5**	**18.0**
Phone CTC	Big	9.6	16.0

Table 3 compares the performance of our 2000-hour word CTC model with the big LM results from the 2000-hour phone CTC model. We observe that our 2000-hour word CTC system

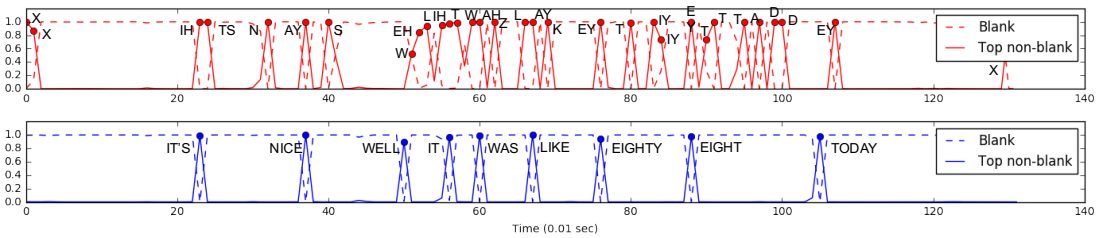

Figure 2: *This figure shows the 2000-hour CTC phone (red, top) and CTC word (blue, bottom) model posteriors for an utterance from the Switchboard test set, along with the phones and words in the respective 1-best paths. "X" denotes the silence phone.*

achieves a WER of 13.0%/18.8% on Switchboard/CallHome without using any LM, while the best phone CTC system using the big LM achieves 9.6%/16.0%. This result is encouraging since recognizing speech using the word CTC model uses no LM or decoder, and hence takes significantly less time and memory compared with the phone CTC model. It is interesting to note that the gains in WER upon increasing the amount of training audio from 300 to 2000 hours is bigger for the word CTC model (7.8%/11.6%) compared to the phone CTC model (4.3%/8.7%). This is due to word targets receiving more training data, and also due to the 1% reduction in OOV rate upon increasing the number of word targets from 10,000 to 25,000.

3.4. Word CTC LM Rescoring

To gauge the impact of using a LM, we rescored the output of the word CTC model with the big LM. The word CTC model posteriors are very spiky, as shown in Figure 2, with the top-scoring word or blank symbol getting most of the probability mass at each frame and the blank symbol dominating an overwhelming majority of frames. We picked the time locations with words as the top hypotheses, and took the next best $K-1$ words at these frames. This gave us a consensus network or sausage-like lattice containing as many nodes as the number of words in the 1-best hypotheses. We also subtracted the word log-priors computed over the training set from the acoustic log-posterior score at each arc to give acoustic log-likelihood.

Table 3 shows the WERs after rescoring the word CTC lattices with the big LM for $K = 10$. Higher values of K did not give more gains. We observe a 0.5%/0.8% absolute improvement in WER, which is similar to what we got upon using the big LM to decode phone CTC models in Table 1. However, the remaining 2.9%/2% gap between the rescored word model and the phone CTC model indicates that some work needs to be done on improving the word CTC model itself. The oracle WERs of the Switchboard and CallHome word CTC lattices was 6.8% and 11.4%, which indicates that the CTC word system can potentially achieve a significantly lower WER.

3.5. Comparison of Word and Phone CTC System Errors

As an initial comparison between the word and phone CTC models, we wanted to see if there is any big discrepancy between the two based on the type of errors committed. Table 4 shows the substitution, deletion, and insertion rates for the 2000-hour phone CTC and word CTC models (with/without LM rescoring). Word CTC lags behind the phone CTC model most often in the deletion rate. With the exception of the unrescored word CTC model, the substitution and insertion rates for the phone and word CTC models differ by $< 1\%$.

Table 4: *This table shows substitution, deletion, and insertion rates of the 2000-hour word and phone CTC systems on the Switchboard and CallHome test sets. Numbers in parentheses show absolute differences with respect to the phone CTC model.*

Model	*SWB Sub.*	*SWB Del.*	*SWB Ins.*
Phone CTC	5.8	2.7	1.1
Word CTC	**7.4 (+1.6)**	**4.1(+1.4)**	1.6 (+0.5)
+LM resc.	6.4 (+0.6)	**4.6 (+1.9)**	1.5 (+0.4)
Model	*CH Sub.*	*CH Del.*	*CH Ins.*
Phone CTC	9.6	4.7	1.7
Word CTC	10.3 (+0.7)	**6.4 (+1.7)**	2.1(+0.4)
+LM resc.	10.0 (+0.4)	**6.1 (+1.4)**	1.9 (+0.2)

We also did a per-utterance comparison of the errors committed by the 2000-hour word and phone CTC systems. For 61.9%/61.1% of utterances from the Switchboard/CallHome test sets, the word and phone CTC systems commit the same number of errors. They commit identical errors in 42%/41% of the utterances. The word CTC system makes at most (less than or equal to) the same number of errors as the phone CTC system in 72.1%/74.9% of the utterances. Hence for a significant number of utterances, the word system makes equal or fewer errors than the phone system, and is much faster to decode with.

4. Conclusion and Future Work

This paper presented the first results using direct acoustics-to-word CTC models on the well-benchmarked Switchboard and CallHome data sets. We presented two techniques for initializing the word CTC models and improving their performance - (1) using the phone CTC BLSTM and hierarchical CTC, and (2) using GloVe word embeddings. Our 2000-hour word CTC system achieved a WER of 13.0%/18.8% on the Switchboard/CallHome data sets, compared to 9.6%/16.0% for the phone CTC system. We also showed the impact of lattice rescoring on the word CTC model. We noted that the word CTC system is at least as good as the phone CTC system in around 70% of the utterances for both the test sets, while being much faster to decode with due to lack of a LM. Future work will focus on improving the word CTC model's performance without scaling the amount of training data by several orders of magnitude, exploring techniques for speeding-up the training of word CTC models further, and evaluating word CTC models for other languages and tasks, especially low-resource ones.

5. References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. 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," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L. Lim, B. Roomi, and P. Hall, "English conversational telephone speech recognition by humans and machines," *arXiv preprint arXiv:1703.02136*, 2017.

[3] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "The Microsoft 2016 conversational speech recognition system," in *Proc. ICASSP*, 2017, pp. 5255–5259.

[4] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Černockỳ, "Empirical evaluation and combination of advanced language modeling techniques," in *Proc. Interspeech*, 2011.

[5] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One billion word benchmark for measuring progress in statistical language modeling," in *Proc. Interspeech*, 2014.

[6] S. F. Chen, "Shrinking exponential language models," in *Proc. HLT-NAACL*, 2009, pp. 468–476.

[7] F. Jelinek, *Statistical methods for speech recognition*. MIT press, 1997.

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, pp. 369–376.

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[10] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. ICASSP*, 2016, pp. 4945–4949.

[11] A. L. Maas, Z. Xie, D. Jurafsky, and A. Y. Ng, "Lexicon-free conversational speech recognition with neural networks," in *Proc. HLT-NAACL*, 2015, pp. 345–354.

[12] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks." in *ICML*, vol. 14, 2014, pp. 1764–1772.

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

[14] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *Proc. ASRU*, 2015, pp. 167–174.

[15] Y. Miao, M. Gowayyed, X. Na, T. Ko, F. Metze, and A. Waibel, "An empirical exploration of CTC acoustic models," in *Proc. ICASSP*, 2016, pp. 2623–2627.

[16] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Proc. Interspeech*, 2015.

[17] G. Zweig, C. Yu, J. Droppo, and A. Stolcke, "Advances in all-neural speech recognition," in *Proc. ICASSP*, 2017, pp. 4805–4809.

[18] H. Soltau, H. Liao, and H. Sak, "Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition," *arXiv preprint arXiv:1610.09975*, 2016.

[19] S. Fernandez, A. Graves, and J. Schmidhuber, "Sequence labelling in structured domains with hierarchical recurrent neural networks," in *Proc. IJCAI*, 2007, pp. 774–779.

[20] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global Vectors for Word Representation," in *Proc. EMNLP*, vol. 14, 2014, pp. 1532–1543.

[21] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[22] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A Matlab-like Environment for Machine Learning," in *Proc. BigLearn, NIPS Workshop*, 2011.

[23] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "CUDNN: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[24] "Warp CTC," https://github.com/baidu-research/warp-ctc.

[25] K. Rao and H. Sak, "Multi-accent speech recognition with hierarchical grapheme based models," in *Proc. ICASSP*, 2017, pp. 4815–4819.

[26] K. Audhkhasi, A. Sethy, and B. Ramabhadran, "Semantic word embedding neural network language models for automatic speech recognition," in *Proc. ICASSP*, 2016, pp. 5995–5999.

[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.