NO NEED FOR A LEXICON? EVALUATING THE VALUE OF THE PRONUNCIATION LEXICA IN END-TO-END MODELS

Tara N. Sainath, Rohit Prabhavalkar, Shankar Kumar, Seungji Lee, Anjuli Kannan, David Rybach
Vlad Schogol, Patrick Nguyen, Bo Li, Yonghui Wu, Zhifeng Chen, Chung-Cheng Chiu

Google, Inc., New York, NY, USA

{tsainath,prabhavalkar,shankarkumar,leesj,anjuli,rybach}@google.com
{vlads,drpng,boboli,yonghui,zhifengc,chungchengc}@google.com

ABSTRACT

For decades, context-dependent phonemes have been the dominant sub-word unit for conventional acoustic modeling systems. This status quo has begun to be challenged recently by end-to-end models which seek to combine acoustic, pronunciation, and language model components into a single neural network. Such systems, which typically predict graphemes or words, simplify the recognition process since they remove the need for a separate expert-curated pronunciation lexicon to map from phoneme-based units to words. However, there has been little previous work comparing phoneme-based versus grapheme-based sub-word units in the end-to-end modeling framework, to determine whether the gains from such approaches are primarily due to the new probabilistic model, or from the joint learning of the various components with grapheme-based units.

In this work, we conduct detailed experiments which are aimed at quantifying the value of phoneme-based pronunciation lexica in the context of end-to-end models. We examine phoneme-based end-to-end models, which are contrasted against grapheme-based ones on a large vocabulary English Voice-search task, where we find that graphemes do indeed outperform phonemes. We also compare grapheme and phoneme-based approaches on a multi-dialect English task, which once again confirm the superiority of graphemes, greatly simplifying the system for recognizing multiple dialects.

1. INTRODUCTION

Traditional automatic speech recognition (ASR) systems are comprised of an acoustic model (AM), a language model (LM) and a pronunciation model (PM), all of which are independently trained on different datasets. AMs take acoustic features and predict a set of sub-word units, typically context-dependent or context-independent phonemes. Next, a hand-designed lexicon (i.e., PM) maps a sequence of phonemes produced by the acoustic model to words. Finally, the LM assigns probabilities to word sequences.

There have been many attempts in the community to fold the AM and PM into one component [1, 2]. This is particularly helpful when training multi-lingual systems [3], as a single AM+PM can be potentially used for all languages. A recent popular approach to jointly learn the AM+PM is to have a model directly predict graphemes. However, to date, most grapheme-based systems do not outperform phone-based systems [4, 5, 6].

More recently, there has been a growing popularity in end-to-end systems, which attempt to learn the AM, PM and LM together in one system. Most work to date has explored end-to-end models which predict either graphemes or wordpieces [7, 8, 9, 10], which removes the need for a hand-designed lexicon as well. These end-to-end systems outperform systems which learn only an AM+PM jointly [11], though to date many of these systems still do not outperform conventional models trained with separate AM, PM and LMs.

This leads to the natural question: how do end-to-end models perform if we incorporate a separate PM and LM into the system? This question can be answered by training an end-to-end model to predict phonemes instead of graphemes. The output of the end-to-end model must then be combined with a separate PM and LM to decode the best hypotheses from the model. End-to-end phoneme models have been explored for a small-footprint keyword spotting task [12], where the authors found that models trained to predict phonemes were better than graphemes. However, this system produced a small number of keyword outputs, thus requiring a simple lexicon and no language model. In our previous work, we also demonstrated that phoneme-based end-to-end systems can be used to improve performance by rescoring lattices decoded from conventional ASR systems [13]. To our knowledge, the present work is the first to explore end-to-end systems trained with phonemes for a large vocabulary continuous speech recognition (LVCSR) task, *where models are directly decoded in the first-pass*.

Our first set of experiments, conducted on a 12,500-hour English Voice Search task, explore the behavior of end-to-end systems trained to predict graphemes vs. phonemes. Our experiments show that the performance of grapheme systems is slightly better than phoneme systems. Since a benefit of end-to-end systems arises in systems trained for multiple dialects/languages, we extend our comparison towards a multi-dialect system trained on 6 different English dialects. Again, we find the grapheme system outperforms the phoneme system.

The rest of this paper is structured as follows. In Section 2 we describe training and decoding an end-to-end model with phonemes. The experimental setup is described in Section 3 and results are presented in Section 4. Finally, Section 5 concludes the paper and discusses future work.

2. INCORPORATING PHONEMES INTO AN END-TO-END MODEL

2.1. Components of Conventional ASR System

Given an input sequence of frame-level features (e.g., log-mel-filterbank energies), $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$, and an output sequence of sub-word units (e.g., graphemes, or phonemes), $\mathbf{y} = \{y_1, y_2, \ldots y_N\}$, the goal of any speech recognition system is to model the distribution over output sequences conditioned on the input, $P(\mathbf{y}|\mathbf{x})$. Typically, the process of finding the best set of recognized words in the network is represented as a composition of finite state transducers (FSTs) [14], shown by Equation 1.

$$D = C \circ L \circ G \tag{1}$$

An acoustic model is trained to map the input $\mathbf{x}$ to a set of context-dependent phones. With reference to Equation 1, a C transducer maps context-dependent phones to context-independent phones (CIPs). The output of C is composed with a pronunciation model, represented by an L transducer. The L transducer takes sequences of CIPs and maps them to words. Finally, the language model is represented by G, which assigns probablities to sequences of words. A potential drawback with this approach is that the acoustic, pronunciation and language models are all trained separately. Furthermore, L is manually curated and a challenging text normalization step is required to map between the verbalized and written representations.

2.2. End-to-end models

End-to-end models attempt to fold parts of the recognition process in Equation 1 into a single neural network. While there are many end-to-end models that have been explored, in this paper we will focus on attention-based models, such as Listen, Attend and Spell (LAS) [8]. This model consist of 3 modules as shown in Figure 1.

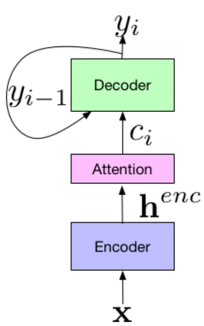

Fig. 1: Components of the LAS end-to-end model.

The *listener* module, also known as the encoder, takes the input features, $\mathbf{x}$, and maps this to a higher order feature representation $\mathbf{h}^{enc}$. We can think of the encoder as similar to a typical acoustic model. The output of the encoder is passsed to an *attender*, which acts like an alignment mechanism, determining which encoder features in $\mathbf{h}^{enc}$ should be attended to in order to predict the next output symbol, y_i. The output of the attention module is passed to the *speller* (i.e., decoder), which takes the attention context, c_i, as well as an embedding of the previous prediction, y_{i-1}, in order to produce a probability distribution, $P(y_i|y_{i-1}, \ldots, y_0, \mathbf{x})$, over the current sub-word unit, y_i, given the previous units, $y_{i-1}, \ldots, y_0$, and input, $\mathbf{x}$. We can think of the decoder as similar to a language model. The model also contains two additional symbols, namely a `<sos>` token which is input to the decoder at time step y_0, indicating the start of sentence and an `<eos>` to indicate end of sentence. The model is trained to minimize the cross-entropy loss on the training data.

2.3. Grapheme Units

Graphemes are a very common subword unit for end-to-end models. In our work, the grapheme inventory includes the 26 lower-case letters a–z, the numerals 0-9, a label representing `<space>`, and punctuation.

The decoding process involves finding the best grapheme sequence, $\mathbf{y}^*$, under the model distribution, in other words:

$$\mathbf{y}^* = \arg\max_y p(\mathbf{y}|\mathbf{x}) = \arg\min_y - \log p(\mathbf{y}|\mathbf{x}) \tag{2}$$

Typically, decoding using an end-to-end model is performed using a beam search. At each step in the beam search, candidate hypotheses are formed by extending each hypothesis in the beam by one grapheme unit. These updated hypotheses are scored with the LAS model, and generally a small number (e.g., 8) of top-scoring candidates are kept to form a new beam for the next decoding step. The model stops decoding when the `<eos>` symbol is predicted.

The prediction of graphemes allows us to remove the need for both the C and L transducers during decoding. This is because the graphemes that are produced by the beam search can simply be concatenated into words, with the predicted `<space>` token indicating word boundaries.

2.4. Phoneme Units

Instead of having the end-to-end model predict graphemes, the model can predict phonemes, but at the cost of requiring additional transducers during decoding which are not needed by the grapheme system. In this work, we explore having the model predict context-independent phonemes (CIP), thus removing the need for a C transducer. Following the small-footprint keyword spotting end-to-end work in [12], we train our model to predict a set of 44 CI phonemes, as well as an extra `<eow>` token, specifying the end of a word, analogous to the `<space>` token in graphemes (e.g., `the cat` $\rightarrow$ `d ax <eow> k ae t <eow>`).

Because of the homophone issue with phonemes (e.g., phoneme `ey` can map to the words 'I' or 'eye'), using a language model, G, is critically important. There are two ways we can incorporate L and G during decoding, mapping from a sequence of phonemes to a sequence of words: first, similar to graphemes, the output of the beam search can produce an n-best list of phonemes; each such phoneme sequence can be composed independently with L and G to get the n-best list of word sequences. This requires having an external LM weight λ on G in order to balance the scores coming from the end-to-end model relative to the scores from L and G. We will refer to this strategy as *N-best Combination*. As an alternative, we can bias each step of the beam search with $L \circ G$, similar to what was done in [15, 16] for graphemes. This strategy, which we will refer to as *Beam-search Combination* is denoted by Equation 3.

$$\mathbf{y}^* = \arg\min_y - \log p(\mathbf{y}|\mathbf{x}) - \lambda \log p_{LM}(\mathbf{y}) - \eta \texttt{coverage} \tag{3}$$

In this equation $p(\mathbf{y}|\mathbf{x})$ is the score from the LAS model, which is combined with a score coming from $L \circ G$ ($p_{LM}(\mathbf{x})$) weighted by an LM weight λ, and a `coverage` term to promote longer transcripts [15] and weighted by η. The benefit of this approach is that the L and G bias each step of the beam search rather than at the end, which is similar to our conventional models. However, one drawback is that as [15] indicates, the equation is a heuristic to combine independent models which can become quite challenging if the end-to-end model term $-\log p(\mathbf{y}|\mathbf{x})$ becomes over-confident, in which case, the weight from the LM component will be ignored. We can also apply $L \circ G$ in both the beam-search and n-best, which will be explored as well.

3. EXPERIMENTAL DETAILS

Our intial experiments are conducted on a $\sim$12,500 hour training set consisting of 15M US English utterances. The training utterances are anonymized and hand-transcribed, and are representative of Google's voice search traffic. This data set is created by artificially corrupting clean utterances using a room simulator, adding varying degrees of noise and reverberation such that the overall SNR is between 0dB and 30dB, with an average SNR of 12dB [17]. The noise sources are drawn from YouTube videos and daily life noisy environmental recordings. We report results on a set of ~14,800 anonymized, hand-transcribed Voice Search utterances extracted from Google traffic.

We also conduct experiments on 5 different English dialects, namely India (IN), Britain (GB), South Africa (ZA), Nigeria & Ghana (NG) and Kenya (KE). A single multi-dialect model is trained on these languages, totaling about 20M utterances (27,500 hours). Noise is artificially added to the clean utterances using the same procedure for US English. We report results on a dialect-specific test set, which is around 10K utterances per test set. We refer the reader to [18] for more details about the experimental setup.

All English experiments use 80-dimensional log-mel features, computed with a 25-ms window and shifted every 10ms. Similar to [19, 20], at the current frame, t, these features are stacked, with 3 frames to the left (for US English) and 7 frames for multi-dialect, and downsampled to a 30ms frame rate. The encoder network architecture consists of 5 unidirectional long short-term memory [21] (LSTM) layers, with the size specified in the results section. Additive attention is used for all experiments [22]. The decoder network is a 2 layer LSTM with 1,024 hidden units per layer. The grapheme systems use 74 symbols while the phoneme systems use 45 CIP for US English, and a unified set of 50 CIPs for multi-dialect.

All neural networks are trained with the cross-entropy criterion, using asynchronous stochastic gradient descent (ASGD) optimization [23] with Adam [24] and are trained using TensorFlow [25].

4. RESULTS

4.1. Tuning CIP Models

Our first set of experiments explore what parameters are important for decoding an end-to-end model trained with CIP.

4.1.1. Tuning LM Weight

First, we explore the behavior of the language model weight (LMW), λ, when $L \circ G$ is incorporated using *N-best Combination* following beam search. Figure 2 shows the WER as a function of the LMW. The figure indicates that WER is heavily affected by the choice of LMW, which seems to be best around 0.1. This also indicates a drawback of using phonemes, namely an external weight needs to be tuned to balance the scores coming from the end-to-end model relative to $L \circ G$. This can be a drawback if the end-to-end model is overconfident and produces a high probability, thus de-emphazising the score from the language model component.

4.1.2. Incorporating End-of-Word Symbol

Next, we explore different ways of using the `<eow>` symbol during decoding. In [12], the `<eow>` symbol was required during decoding and was shown to help in identifying the spacing between words. In particular, since models were decoded without a separate LM, requiring an `<eow>` symbol between words was found to be critical to minimize false positives (e.g., to avoid false triggering on the phrase `America`, for the keyword `Erica`). However, in a LVCSR task like Voice Search where we use a separate PM and LM for L and G, requiring `<eow>` might cause the model to make errors and predict

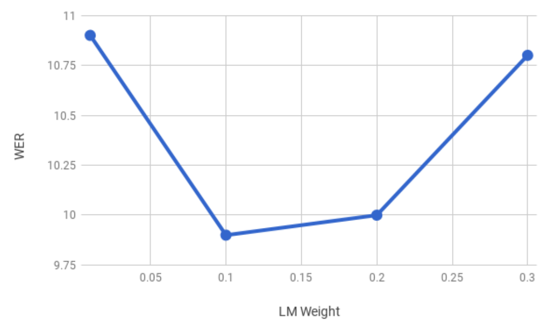

Fig. 2: WER as a function of LM weight.

incorrect words if `<eow>` is not correctly predicted. Table 1 shows that it is better to make `<eow>` optional rather than required. Note for these experiments, $L \circ G$ is incorporated using *N-best Combination*.

System	WER
LAS unid, required `<eow>`	10.2
LAS unid, optional `<eow>`	9.7

Table 1: WER Phoneme `<eow>` analysis. Because it is hard to predict `<eow>`, it is better to make it optional.

4.1.3. Where to use $L \circ G$

Finally, we study where to apply $L \circ G$, specifically if it should be applied either during the beam search *Beam-search Combination*, following the beam search *N-best Combination*, or in both places. Applying $L \circ G$ in both places requires tuning two separate LMW weights, for the G during and after the beam search.

Figure 3 shows the WER of the final system as the beam search LMW is increased from 0.0 to 0.1. For illustrative purposes, we set the weight of the first and second LM to sum to 0.1, though more extensive sweeping of both LMWs did not improve performance further. The figure shows that a slight improvement is obtained with *N-best Combination* (9.7, LMW= 0.0) compared to *Beam-search Combination* (9.8, LMW=0.1). This illustrates that the decoder of the LAS model is strong enough to learn the correct phone sequence, and thus *N-best Combination* is sufficient enough to yield reasonable results. The rest of the results in this paper are reported with *N-best Combination*.

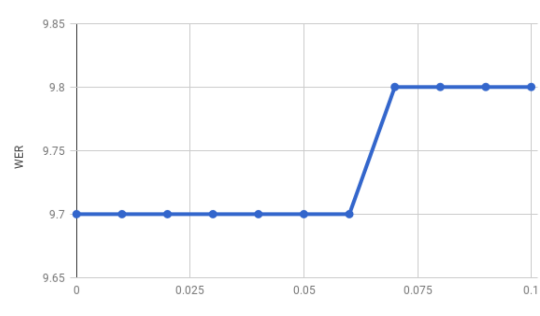

Fig. 3: WER as a function of beam-search LM Weight.

4.2. Phoneme vs. Grapheme Comparison, Model Architecture

Having established a good recipe for training with CIP, we now compare the performance of phoneme and grapheme systems for different LAS architectures, namely a single head unidirectional LAS system and a multi-head unidirectional LAS system. We are specifically interested in multi-head (MHA) for LAS, as MHA has been shown to give state-of-the-art performance for LAS grapheme systems [26]. MHA allow the end-to-end model to jointly attend to information at different positions in the encoder space with multiple attention heads. The single head model is a 5x1024 encoder with 1 attention head and 2x1024 decoder. The MHA model is a 5x1400 encoder with 4 attention heads, followed by a 2x1024 decoder.

Table 2 indicates that for single-head attention, both phoneme and grapheme systems have similar performance. However, for MHA the phoneme system lags behind the grapheme system. One hypothesis we have is as the end-to-end model, including the encoder and decoder, gets stronger, from single to multi-head attention, having a model which jointly integrates the AM, PM and LM (i.e., training with graphemes) is better than separate integration (i.e., training with phonemes).

Model	Phoneme	Grapheme
unid LAS	9.7	9.8
MHA LAS	8.6 (1.4/1.8/5.4)	8.0 (1.1/1.3/5.6)

Table 2: CIP vs. Graphemes Across Model Architectures. The (del/ins/sub) is indicated in parenthesis.

To understand the errors made by phonemes and graphemes, we pulled a few representative examples. Table 3 shows examples of where the grapheme system wins over the phoneme system. The first example in the table indicates that the phoneme system has slightly higher deletions, likely because of the incorporation of the external L and G and the need to tune an LMW. This is also confirmed quantitiatively by looking at the deletions, insertions, and substitutions in Table 2. In addition, because of the hand-designed lexicon L, the second example shows that phoneme system does not do as well with text normalization. Finally, the grapheme system benefits from making a joint decision for disambiguating homophones while the chained phoneme system does not, as shown in the third example.

Grapheme	Phoneme
let me see a clown	Let me see
How old is 50 cents	How old is **$0.50**
Easy Metallica songs to play on the guitar	**AZ** Metallica songs to play on the guitar

Table 3: Grapheme Wins. Phoneme errors indicated in **red**.

In contrast, Table 4 gives examples where the phoneme system wins over the grapheme system. The phoneme system wins on proper nouns and rare words, aided by the hand designed lexicon L and the LM G, which is trained on a billion word text-only corpora.

Grapheme	Phoneme
Albert Einstein versus **Singapore**	Albert Einstein vs. Stephen Hawking
Head Start on **Concord** New York	Head Start Ronkonkoma New York
Charles **Lindberg** in Paris	Charles Lindbergh in Paris

Table 4: Phoneme Wins. Grapheme errors indicated in **red**.

4.3. Comparison for Multi-dialect

In this section, we compare the performance of phones vs. graphemes for a multi-dialect English system. Both systems use a 5x1024 encoder with single-head attention, followed by a 2x1024 decoder. For the phoneme systems, we use a unified phone set and a unified L, but a language-specific G for each language. The results are shown in Table 5. The table shows that across the board the phoneme system is worse than the grapheme system.

Table 6 gives a few examples of where the phoneme system makes errors compared to the grapheme system. In addition to text norm and deletion errors like in English, the multi-dialect phone system also has many pronunciation errors. The table illustrates this the disadvantage of having a hand-designed lexicon. Overall, a grapheme end-to-end model provides a much simpler and more effective strategy for multi-dialect ASR.

Dialect	IN	GB	ZA	NG	KE
grapheme	18.4	14.1	13.8	34.5	19.9
phoneme	31.6	18.1	18.6	39.0	24.8

Table 5: WER of CIP vs. Graphemes For Mulit-dialect

Grapheme	Phoneme
Chris Moyles bake off	Chris **Miles** bake off
Ukip Sussex candidates	**You** kip Sussex candidates
What does Allison mean	What does **Alison** mean
My name is Reese	My name is **Rhys**

Table 6: Grapheme Wins for Multi-dialect. Phoneme errors indicated in **red**.

5. CONCLUSIONS

In this paper, we examined the value of a phoneme-based pronunciation lexica in the context of end-to-end models. Specifically, we compared using phone vs. grapheme systems with an end-to-end attention-based model. We found that for both US English and multi-dialect English, the grapheme systems were superior to the phone systems. Error analysis shows that the grapheme systems lose on proper nouns and rare words, where the hand-designed lexica help. Future work will look at combining the strengths of both of these units into one system.

6. ACKNOWLEDGEMENTS

The authors would like to thank Eugene Weinstein and Michiel Bacchiani for useful discussions. In addition, thanks to Alyson Pitts, Jeremy O'Brien, Shayna Lurya and Evan Crewe for help with the multi-dialect experiments.

7. REFERENCES

[1] I. McGraw, I. Badr, and J. R. Glass, "Learning lexicons from speech using a pronunciation mixture model," in *IEEE Transactions on Audio, Speech, and Language Processing*, 2013, vol. 21, p. 357366.

[2] L. Lu, A. Ghoshal, and S. Renals, "Acoustic data-driven pronunciation lexicon for large vocabulary speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, 2013, pp. 374–379.

[3] S. Kanthak and H. Ney, "Multilingual acoustic modeling using graphemes," in *Proceedings of European Conference On Speech Communication and Tenchology*, 2003, pp. 1145–1148.

[4] Y. Sung, T. Hughes, F. Beaufays, and B. Strope, "Revisiting graphemes with increasing amounts of data," in *ICASSP*, 2008.

[5] F. Eyben, M. Wollmer, B. Schuller, and A. Graves, "From speech to letters using a novel neural network architecture for grapheme based ASR," in *Automatic Speech Recognition and Understanding, ASRU*, 2009.

[6] K. Rao and H. Sak, "Multi-accent Speech Recognition with Hierarchical Grapheme Based Models," in *Proc. ICASSP*, 2017.

[7] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition," in *Proc. NIPS*, 2015.

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

[9] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *arXiv preprint arXiv:1512.02595*, 2015.

[10] K. Rao, R. Prabhavalkar, and H. Sak, "Exploring Architectures, Data and Units for Streaming End-to-End Speech Recognition with RNN-Transducer," in *Proc. ASRU*, 2017.

[11] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A Comparison of Sequence-to-sequence Models for Speech Recognition," in *Proc. Interspeech*, 2017.

[12] Y. He, R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin, and I. McGraw, "Streaming Small-footprint Keyword Spotting Using Sequence-to-Sequence Models," in *Proc. ASRU*, 2017.

[13] R. Prabhavalkar, T. N. Sainath, B. Li, K. Rao, and N. Jaitly, "An Analysis of "Attention" in Sequence-to-Sequence Models," in Proc. Interspeech," in *Proc. Interspeech*, 2017.

[14] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Speech recognition with weighted finite-state transducers," in *Handbook of Speech Processing*, Jacob Benesty, M. Sondhi, and Yiteng Huang, Eds., chapter 28, pp. 559–582. Springer, 2008.

[15] J. Chorowski and N. Jaitly, "Towards Better Decoding and Language Model Integration in Sequence to Sequence Models," in *Proc. Interspeech*, 2017.

[16] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *submitted to Proc. ICASSP*, 2018.

[17] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. N. Sainath, and M. Bacchiani, "Generated of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in google home," in *Proc. Interspeech*, 2017.

[18] B. Li, T. N. Sainath, K. Sim, M. Bacchiani, E. Weinstein, P. Nguyen, Z. Chen, Y. Wu, and K. Rao, "Multi-dialect speech recognition with a single sequence-to-sequence model," in *submitted to Proc. ICASSP*, 2018.

[19] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition," in *Proc. Interspeech*, 2015.

[20] G. Pundak and T. N. Sainath, "Lower Frame Rate Neural Network Acoustic Models," in *Proc. Interspeech*, 2016.

[21] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *CoRR*, vol. abs/1409.0473, 2014.

[23] J. Dean, G.S. Corrado, R. Monga, K. Chen, M. Devin, Q.V. Le, M.Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A.Y. Ng, "Large Scale Distributed Deep Networks," in *Proc. NIPS*, 2012.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of ICLR*, 2015.

[25] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," Available online: http://download.tensorflow.org/paper/whitepaper2015.pdf, 2015.

[26] C. Chen, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, N. Jaitly, B. Li, and J. Chorowski, "State-of-the-art speech recognition with sequence-to-sequence models," in *submitted to Proc. ICASSP*, 2018.