AN ANALYSIS OF INCORPORATING AN EXTERNAL LANGUAGE MODEL INTO A SEQUENCE-TO-SEQUENCE MODEL

Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N. Sainath,
Zhifeng Chen, Rohit Prabhavalkar

Google, Inc., USA

{anjuli,yonghui,drpng,tsainath,zhifengc,prabhavalkar}@google.com

ABSTRACT

Attention-based sequence-to-sequence models for automatic speech recognition jointly train an acoustic model, language model, and alignment mechanism. Thus, the language model component is only trained on transcribed audio-text pairs. This leads to the use of *shallow fusion* with an external language model at inference time. Shallow fusion refers to log-linear interpolation with a separately trained language model at each step of the beam search. In this work, we investigate the behavior of shallow fusion across a range of conditions: different types of language models, different decoding units, and different tasks. On Google Voice Search, we demonstrate that the use of shallow fusion with an neural LM with wordpieces yields a 9.1% relative word error rate reduction (WERR) over our competitive attention-based sequence-to-sequence model, obviating the need for second-pass rescoring.

1. INTRODUCTION

Sequence-to-sequence models have started to gain popularity for automatic speech recognition (ASR) tasks, particularly for their benefit of folding various parts of the speech recognition pipeline (i.e., acoustic, prononcuation and language modeling) into one neural network [1, 2, 3, 4]. For example, the Listen, Attend, and Spell (LAS) model jointly learns an encoder, which serves as an acoustic model, a decoder, which serves as a language model (LM), and an attention mechanism, which learns alignments. Recently, a comparison of these different methods showed that performance still lagged behind a state-of-the-art ASR system with separate acoustic, pronunciation and language models [5]. The focus of this paper is to explore a means of making LAS competitive to a conventional ASR model.

We propose that one reason for the performance degradation could be that the LAS decoder, which replaces the LM component in a traditional ASR system, is trained only on transcribed audio-text pairs, which is about 15 million utterances for the Google Voice Search task [5]. In comparison, state-of-the-art LMs are typically trained on a billion words or more [6]. This raises the question of whether the LAS decoder can learn a strong enough LM from the training transcripts. In particular, we posit that in a task like Google Voice Search, which has a very long tail of queries, the training transcripts may not sufficiently expose the LAS decoder to rare words and phrases.

However, these words may appear in auxiliary sources of text-only data such as web documents or news articles, which comprise billions of words. This work investigates the impact of training a separate LM on auxiliary text-only data, and incorporating this model as an additional cost term when decoding a LAS model.

Several recent works have also investigated the use of LMs with attention-based models. [1] demonstrated significant improvement by rescoring the n-best hypotheses produced by LAS with a 5-gram LM. [2] extended this idea by performing log-linear interpolation between LAS and an n-gram LM at each step of the beam search, a method we will henceforth refer to as *shallow fusion*, following the terminology of [7]. Shallow fusion was further studied in [8], which extended it with use of a coverage penalty. Both of these works were limited to Wall Street Journal (WSJ), which, given its scarcity of data, stands to gain more from an external LM than a large-scale task such as Google Voice Search. All of these works only investigated n-gram LMs, and all focused on bidirectional models that output graphemes.

The use of an external LM has also been investigated in the context of training, such that the LAS model could learn when and how to use the LM [7, 9, 10, 11]. These works applied Recurrent Neural Network (RNN) LMs, but this was largely cited as a means to make the integration simpler. None provided a direct comparison of RNN LMs to n-gram LMs. Further, they were all limited to grapheme systems, with [7] and [9] focused on machine translation.

This work has two goals. First, we extend the work of [8] by exploring the behavior of shallow fusion across different sub-word units and different types of LMs on a small corpus task. We find that RNN LMs are more effective at reducing error than n-gram LMs, with the magnitude of this reduction consistent across sub-word units.

The second goal of our work is to explore the behavior of shallow fusion on a large-scale, large-vocabulary English Voice Search task. Voice Search has much more training data than WSJ so it is not clear that the benefits observed on WSJ should necssarily translate; given sufficient training data, the LAS decoder may be strong enough to eliminate the effect of any external LM. Additionally, Voice Search requires a unidirectional model, which has not previously been studied with shallow fusion. Ultimately, we find that shallow fusion with a worpiece-level RNN LM yields a 9.1% relative WERR on a competitive unidirectional baseline.

The next two sections will provide more details about the method we use for integrating the LM and the variants that we compare. Section 4 describes the setup for our experiments on two different tasks, and Section 5 provides the results of these experiments. Finally, in Section 6 we conclude this study.

2. SHALLOW FUSION WITH LAS MODELS

2.1. Listen, attend, and spell

As shown inside the dotted line box in Figure 1, the LAS model consists of an encoder ("listen"), an attention mechanism ("attend"),

and a decoder ("spell").

The encoder, which is akin to an acoustic model, consists of a stack of long short-term memory layers (LSTMs) [12]. These take as input a sequence of d-dimensional feature vectors, $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathbb{R}^d$, and produces a higher-order feature representation, denoted $\mathbf{h}_1^{\text{enc}}, \cdots, \mathbf{h}_T^{\text{enc}}$.

The output of the encoder is passed to an attention mechanism, which determines which part of the encoder features to attend to in order to predict each output symbol, effectively performing a dynamic time warping. The output of the attention mechanism is a single context vector that encodes this information.

Finally, the decoder is another stack of LSTMs which is conditioned on the context vector. Given the context vector and the previous prediction y_{u-1} at timestep u, the decoder network generates logits $\mathbf{h}^{\text{dec}_u}$. These are passed through a softmax to compute a probability distribution $P(y_u|\mathbf{h}^{\text{dec}_u})$.

The decoder can be thought of as a neural LM conditioned on the acoustic model output; however, since the LAS model is structured such that the encoder feeds the decoder, this internal LM can only be trained on audio-text pairs. In the next section, we will discuss the incorporation of an external LM.

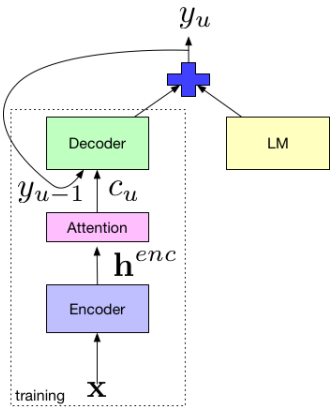

Fig. 1: The dotted line box shows the basic LAS model, including an encoder, attention, and decoder. In shallow fusion, an external LM is incorporated via log-linear interpolation.

2.2. Integrating a language model

Shallow fusion, shown in Figure 1, is a method for incorporating an external LM during inference only. As the figure shows, only the contents of the dotted line box are used to train of the LAS model. Then, at inference time, we perform log-linear interpolation with an LM at each step of the beam search. In other words, while the objective criterion for decoding a sequence-to-sequence model typically would be:

$$\mathbf{y}^* = \arg\max_y \log p(y|x) \tag{1}$$

we instead use the following criterion:

$$\mathbf{y}^* = \arg\max_y \log p(y|x) + \lambda \log p_{LM}(y) + \gamma c(x, y) \tag{2}$$

where p_{LM} is provided by an LM, and λ and γ are tuned on a dev set. $c(x, y)$ is referred to as a *coverage penalty* and is designed to penalize incomplete transcripts. It measures the extent to which the input frames are "covered" by the attention weights, computed as:

$$c(x, y) = \sum_j \log(\min(\sum_i a_{i,j}, 0.5)) \tag{3}$$

where $a_{i,j}$ is attention probability of the jth output label y_j on the ith input feature vector x_i. By promoting transcripts which require attention to more of the audio frames, the coverage penalty addresses the common sequence-to-sequence failure mode of assigning high probability to a truncated output sequence [13]; like [8, 14], however, we apply this only at decoding time. The effect of promoting longer transcripts is similar to that of a length normalization or word insertion reward; unlike these atlernatives, though, it is less prone to produce "babbling", since simply inserting more tokens while attending to the same frames will not reduce the coverage penalty.

An alternative method of incorporating an LM would be to simply rescore the n best transcripts produced by the beam search, as in [1]. Our initial experiments on the WSJ corpus showed this method provided some reduction in error, but not as much as shallow fusion. This is because the correct prefix may get pruned by the beam search early on, and not make it into the n-best list.

3. EXPLORING SHALLOW FUSION ACROSS TASKS, DECODING UNITS, AND TYPES OF LANGUAGE MODELS

3.1. Tasks: WSJ vs. Google Voice Search

This work investigates the impact of shallow fusion on two different tasks. This is because we hypothesize that there are several task-specific properties that can affect the relative gain afforded by an external LM:

- *Size of training corpus*, because on a large training corpus the LAS decoder will itself be a very strong LM.
- *Size of vocabulary*, as some of the benefit of an external LM may simply be exposure to unseen words and phrases.
- *Availability of LM training data*, since the LM training data must come from the same domain as the task

Our first set of experiments focuses on the WSJ corpus for several reasons. First, we have a large amount of text-only data also from WSJ, which reduces the possibility of domain mismatch between the LM and the LAS model. Second, given the relatively small size of the WSJ corpus, we see that indeed many errors in a vanilla LAS model result from a poor LM. Third, we can use the standard setup for the training data and vocabulary of the LM, making comparison to previous works more direct. Thus WSJ serves as a useful testbed for measuring the contribution of an external LM.

On the other hand, the small training corpus, means that the gains seen on the WSJ task may not necessarily transfer to a task with a much larger training set. For this reason our second set of experiments is done on the Google Voice Search task. Two notable properties of Voice Search are that it has a large vocabulary and it has a very long tail of queries.

3.2. Decoding Units: Wordpieces vs. Graphemes

While previous works have only investigated shallow fusion for graphemes, we extend our study to wordpieces. Wordpieces [15] are sub-word units that can be as small as a single grapheme or as large as a complete word. First, a fixed wordpiece vocabulary is determined based on frequencies of words in a training corpus. Once the set of valid wordpieces is learned, a transcript can be tokenized

by choosing the longest possible component wordpieces in a greedy fashion.

Like graphemes, wordpieces have the advantage that there are no out-of-vocabulary terms because any word can be decomposed into wordpieces. (All graphemes are included in the wordpiece vocabulary.) But wordpieces have the additional benefit that they effectively capture more context per decoding step than graphemes. This reduces the length of dependencies that must be learned by an LM.

For example, the phrase "the company announced today" consists of 27 graphemes, which means that a grapheme-level LM (LM-G) would require 27 decoding steps to output the full phrase; but a wordpiece-level LM (LM-WP) might compose this phrase as, for example `the _com pany _announc ed _today` which would require only 5 steps to output. Since it requires fewer steps across which to memorize dependencies, we expect that LM-WP can achieve lower (word-level) perplexity than LM-G, which in turn could make it more effective in shallow fusion.

3.3. Language Models: RNNs vs. n-gram

This work further compares shallow fusion across various types of LMs. Previous works have focused on n-gram LMs when applying shallow fusion [8, 2] or RNN LMs for deep or cold fusion [7, 11]. Here we consider both n-gram LMs and RNN LMs [16] for shallow fusion.

There are several reasons that n-gram LMs have been preferred in past work. First, they can incorporate word-level constraints. Since we incorporate the LM at each step of the beam search, the LM must provide a probability distribution at the level of the LAS model's decoding unit (either grapheme or wordpiece). In the case of an RNN LM, this means that we train at the grapheme or wordpiece level. In the case of an n-gram LM, however, there are two possible setups. The most obvious is to train the LM at the level of the decoding unit (grapheme or wordpiece). But in order to have a strong grapheme-level LM, it is necessary to train at a very high order, such as 20-gram, to capture at least a few words worth of context. Following [2], an alternative is to train the LM at the word level, and then, using the Weighted Finite State Transducer framework [17, 18], compose it with a "speller" which breaks each word into its component units (graphemes or wordpieces). In this way, we can still get a probability distribution at the unit level, while incorporating the knowledge of a word-level LM.

Furthermore, this latter setup implicitly introduces a dictionary. In a task like WSJ, the baseline model has a relatively weak decoder, so it will frequently output sequences of graphemes which do not comprise English words. The dictionary constraints imposed by the n-gram LM can be helpful to prune these out.

Finally, in a task like Google Voice Search, there are many sources of data that can potentially be useful in an external LM. We can use Bayesian interpolation to combine n-gram LMs trained individually on each of these domains, optimizing the interpolation weights against WER on a dev set [19]. Currently this sort of technique only exists for n-gram LMs.

Despite all these advantages of n-gram LMs, recent literature has shown that state-of-the-art RNN LMs have a significantly lower perplexity than n-gram LMs on the 1 billion word benchmark, particularly on rare words [6]. Thus we hypothesize that they should also provide a greater reduction in error when used in shallow fusion. Furthermore, given enough training data, as we have in the Google Voice Search task, we suggest that the introduction of the dictionary may not be necessary; in fact, it may be limiting to the model since the LAS model can actually "sound out" words that it has never seen before but which are spelled phonetically. Though the techniques of Bayesian interpolation and incorporating dictionary constraints currently apply only the n-gram models, we posit that analogous methods should be possible for RNN LMs, and identify these as areas for future work.

4. EXPERIMENTAL DETAILS

4.1. Wall Street Journal

Our experiments are conducted on two tasks. The first is the WSJ dataset. Following the setup in [8], we train on *si284*, validate on *dev93* and evaluate on *eval92*.

For grapheme experiments, our baseline model is a LAS model with 3 convolutional layers and a convolutional LSTM layer, followed by 3 bidirectional [20] LSTM layers. The output vocabulary is 72 graphemes. Temporal label smoothing is applied as described in [8]. For wordpiece experiments, our baseline model has the same architecture as the grapheme model, except that the output vocabulary has 1,024 wordpieces and no label smoothing is applied because label smoothing resulted in a weaker model. Instead, L2 regularization is used. Larger wordpiece vocabularies also resulted in worse models.

The external LMs are trained using the WSJ text corpus and extended vocabulary (approximately 150K terms) provided in the Kaldi WSJ s5 recipe [21]. The RNN LMs consist of two LSTM layers of 512 hidden units. The word- , grapheme- , and wordpiece-level n-gram LMs are all trained with Katz smoothing and pruned to between 15M and 20M n-grams. The word-level LM is composed with a speller to decode at the grapheme or wordpiece level.

4.2. Google Voice Search

The second task is a $\sim$12,500 hour training set consisting of 15M 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. The noise sources are from YouTube and daily life noisy environmental recordings. We report results on two sets of $\sim$14,800 anonymized, hand-transcribed Voice Search utterances each, extracted from Google traffic.

The baseline model for Voice Search experiments has an encoder consisting of 5 unidirectional LSTM layers of 1,400 units each, a decoder consisting of 2 LSTM layers with 1,024 hidden units each, and a multi-headed attention mechanism [22]. We use a unidirectional encoder because the Voice Search task requires a streaming model.

All experiments use 80-dimensional log-mel features, computed with a 25-ms window and shifted every 10ms. Similar to [23, 24], at the current frame, t, these features are stacked with 3 frames to the left and downsampled to a 30ms frame rate. The models are trained with the cross-entropy criterion, using asynchronous stochastic gradient descent optimization in TensorFlow [25].

Our text dataset consists of billions of sentences from several sources: untranscribed anonymized Voice Search queries, untranscribed anonymized voice dictation queries, anonymized typed queries from Google Search, as well as the transcribed training utterances mentioned above. The production LMs denoted as PRODLM1 and PRODLM1 are both 5-gram LMs with a vocabulary of 4M. PRODLM1 is constructed as a Bayesian-interpolated mixture of LMs trained on the individual data sources [19], while PRODLM2 is trained on all data. Following [26], the RNN LM is trained on about half a

billion sentences sampled from the full pool data. It consists of two LSTM layers of 2,048 units each.

5. RESULTS

5.1. Comparing LMs for shallow fusion

We begin by comparing three types of LMs in the context of shallow fusion with the LAS grapheme model LAS-G on the WSJ task: (1) an RNN LM trained on graphemes (RNN-G), (2) a 20-gram LM trained on graphemes (20-GRAM-G), and (3) a 3-gram LM trained on words and composed with a speller (3-GRAM-W).

Comparing these, we see that 3-GRAM-W barely outperforms 20-GRAM-G. This shows that, given the same amount of context, having word constraints and an implicit dictionary has only a slight benefit. RNN-G, however, outperforms both of the n-gram LMs, suggesting while the word constraints may help, they are insufficient to make up the gap between RNN LMs and n-gram LMs. One opportunity for future work would be incorporating word constraints into RNN-G.

System	Dev	Test
LAS-G	13.0	10.3
LAS-G + 20-GRAM-G	10.3	7.7
LAS-G + 3-GRAM-W	10.0	7.6
LAS-G + RNN-G	9.3	6.9

Table 1: WER of LAS-G fused with various LMs. While word constraints do help the n-gram LM, RNN-G performs even better.

5.2. Extending shallow fusion to wordpiece models

Next, we perform a comparison for LAS-WP. Since we have shown that word constraints are helpful for sub-word-level n-gram LMs, we limit our comparison to just two LMs: (1) an RNN LM trained on wordpieces (RNN-WP), and (2) a 3-gram LM trained on words and composed with a speller (3-GRAM-W).

As Table 2 shows, we see the same trend on LAS-WP, with RNN-WP significantly better than 3-GRAM-W. However, it should be noted that the baseline LAS-WP is worse than LAS-G. This is likely due to the small amount of data being insufficient to train the large number of additional parameters: we found that the larger we made the wordpiece vocabulary, the worse the model became. As a result of this difference, the LM results for LAS-WP are not directly comparable to the LM results for LAS-G. The main observation we make is that the RNN performs best in both cases, with the relative improvement being roughly consistent for both graphemes and wordpieces.

System	Dev	Test
LAS-WP	15.7	12.3
LAS-WP + 3-GRAM-W	12.9	9.3
LAS-WP + RNN-WP	11.5	8.2

Table 2: WER of LAS-WP combined with various LMs on WSJ. RNN-WP again performs best.

5.3. Scaling up to Voice Search

We now turn to the Voice Search task. First, since we have an abundance of training data, we see in the first two lines of Table 3 that the wordpiece model (LAS-WP) is now comparable with the grapheme model (LAS-G). Thus our analysis here is limited to LAS-WP.

In the traditional HMM/CTC-based system, the decoding proceeds in two passes: the first pass uses a small n-gram LM (PRODLM1), which fits in memory and minimizes the search space to meet real-time requirements. The first pass generates an N-best list which we rescore with a much larger n-gram LM (PRODLM2) [19]. In the third and fourth lines of Table 3 we see the results of applying the production LMs to the LAS model with shallow fusion: the LM inherent in LAS is quite competitive, but there is a small gain from the highly-pruned PRODLM1. The much larger PRODLM2, despite being 40x larger, provides only slightly more improvement. In addition, PRODLM2 is 80GB and must be run on multiple servers. This is operationally unwieldy and cannot be efficiently integrated with low latency during the first pass.

On the other hand, while computationally expensive, RNN LMs are known to be more compact than their n-gram counterparts. In line 5 of Table 3, LAS-WP + RNN-WP, we show that the shallow fusion of LAS with RNN-WP provides an even greater benefit than PRODLM2. Its much lower memory footprint (1.1 GB) allows it to fit in the first pass. We then rescore the system with PRODLM2 (as LAS-WP + RNN-WP + PRODLM2). This yields no further gain, showing that we have obviated the need for a second-pass rescoring at all.

Thus, as with WSJ, we see that RNN-WP more effectively encodes the LM information compared to the n-gram model. In addition, RNN-WP is 1.5% the size of PRODLM2, and also enjoys the additional benefit of not having out-of-vocabulary words since it is trained on wordpieces. Note that both PRODLM1 and PRODLM2 are interpolated across several data-source-specific LMs, while RNN-WP uses ad hoc mixing weights for the various data sources. Investigating a more principled method of mixing the data sources for RNN-WP is an opportunity for future work.

System	Dev	Test	LM size
LAS-G	9.5	7.7	0GB
LAS-WP	9.2	7.7	0GB
LAS-WP + PRODLM1	8.8	7.4	2GB
LAS-WP + PRODLM2	8.7	7.2	80GB
LAS-WP + RNN-WP	8.4	7.0	1.1GB
LAS-WP + RNN-WP + PRODLM2	8.4	7.0	81.1GB

Table 3: WER of shallow fusion of LAS with production n-gram LMs and an RNN LM. The RNN LM captures all the benefits of PRODLM2 in a compact form.

6. CONCLUSIONS

In this work we investigated the technique of shallow fusion, in which an external LM is used to augment a LAS model at inference time. We demonstrated that on the small WSJ task, an RNN LM yielded greater improvement than an n-gram LM, and the gains were consistent across graphemes and wordpieces. On the much larger Voice Search task, we showed that the decoder LM inherent in LAS is already very competitive, yielding little benefit from shallow fusion with the first-pass production LM. However, we found that shallow fusion with an RNN LM provided greater benefit. In fact, with 9.1% relative WERR on a competitive unidirectional system, it eliminated the need for a second pass rescoring, despite being 70 times smaller than the second pass LM.

7. ACKNOWLEDGEMENTS

The authors would like to thank Jan Chorowski, Navdeep Jaitly, Shankar Kumar, Kanishka Rao, Brian Roark, and David Rybach, for helpful discussions.

8. REFERENCES

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

[2] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-End Attention-based Large Vocabulary Speech Recognition," in *Proc. ICASSP*, 2016.

[3] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, vol. abs/1211.3711, 2012.

[4] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labeling Unsegmented Seuqnece Data with Recurrent Neural Networks," in *Proc. ICML*, 2006.

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

[6] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *CoRR*, vol. abs/1602.02410, 2016.

[7] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H. Lin, F. Bougares, H. Schwenk, and Y.Bengio, "On using monolingual corpora in neural machine translation," *CoRR*, vol. abs/1503.03535, 2015.

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

[9] C. Gulcehre, O. Firat, , K. Xu, K. Cho, and Y. Bengio, "On integrating a language model into neural machine translation," vol. 35, pp. 137–148, 2017.

[10] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm," 2017.

[11] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," *CoRR*, vol. abs/1708.06426, 2017.

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

[13] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," *CoRR*, vol. abs/1601.04811, 2016.

[14] Y. Wu, M. Schuster, and et. al., "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[15] M. Schuster and K. Nakajima, "Japanese and Korean voice search," *2012 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012.

[16] T.Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanbur, "Recurrent neural network based language model," *Proc. Interspeech*, 2010.

[17] M. Mohri, F. Pereira, and M. Riley, "Weighted finite state transducers in speech recognition," vol. 16, pp. 69–88, 2002.

[18] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: A general and efficient weighted finite-state transducer library," pp. 11–23, 2007.

[19] C. Allauzen and M. Riley, "Bayesian language model interpolation for mobile speech input," *Proc. Interspeech*, 2011.

[20] M. Schuster and K. K. Paliwal, "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition," *Artificial Neural Networks: Formal Models and Their Applications-ICANN*, pp. 799–804, 2005.

[21] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, J. Silovsky P. Schwarz, G. Stemmer, , and K. Vesely, "The kaldi speech recognition toolkit," 2011.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017.

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

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

[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] 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.