PHOEBE: PRONUNCIATION-AWARE CONTEXTUALIZATION FOR END-TO-END SPEECH RECOGNITION

Antoine Bruguier, Rohit Prabhavalkar, Golan Pundak, Tara N. Sainath

Google USA

{tonybruguier,prabhavalkar,golan,tsainath}@google.com

ABSTRACT

End-to-End (E2E) automatic speech recognition (ASR) systems learn word spellings directly from text-audio pairs, in contrast to traditional ASR systems which incorporate a separate pronunciation lexicon. The lexicon allows a traditional system to correctly spell rare words observed only in LM training, if their phonetic pronunciation is known during inference. E2E systems, however, are more likely to misspell rare words.

We propose an E2E model which benefits from the best of both worlds: it outputs graphemes, and thus learns to spell words directly, while leveraging pronunciations for words which might be likely in a given context. Our model is based on the recently proposed Contextual Listen, Attend, and Spell (CLAS) model. As in CLAS, our model accepts a set of bias phrases, which are first converted into fixed length embeddings which are provided as additional inputs to the model. Unlike CLAS, which accepts only the textual form of the bias phrases, the proposed model also has access to the corresponding phonetic pronunciations, which improves performance on challenging sets which include words unseen in training. The proposed model provides a 16% relative word-error-rate reduction over CLAS when both the phonetic and written representation of the context bias phrases are used.

Index Terms— speech recognition, sequence-to-sequence, biasing, pronunciation, LAS

1. INTRODUCTION

Traditionally, automatic speech recognition (ASR) systems have relied on three models: an acoustic model (AM), a pronunciation model (PM), and a language model (LM) [1]. These three models are mostly tuned independently using different objective functions. For example, we tune the AM to predict a sequence of context-dependent phones from acoustics and we tune the LM to lower perplexity.

Designs decisions for the PM influence both the AM and the LM. For the PM, we need to choose a set a phonemes to represent all the possible distinct unit of sounds, thus determining the set of labels the AM has to predict. All the words in the LM must have a pronunciation. Moreover, having an accurate PM with a large coverage is a challenging task. For a given variant of a language, the phoneme set varies from region to region. For example, in American English, native speakers do not agree whether words like *pen* and *pin* are pronounced the same way. Another issue is reduced pronunciations, where speakers may or may not pronounce the letter 't' in words like *twenty*. Finally, to have a pronunciation for every word, the model must back off to a grapheme-to-phoneme (G2P) model.

Recently, researchers have investigated end-to-end (E2E) models which predict graphemes or wordpieces [2, 3, 4, 5, 6]. These models fold the AM, PM and LM into a single neural network. Recent work [7, 8] has shown that the joint optimization of these modules not only helps make the ASR model much simpler, but also achieve a competitive word error rate (WER) compared to a conventional ASR system. However, one of the downsides of E2E models is that they perform poorly on infrequent words of a language [9]. Since E2E models require human-transcribed voice data, which is expensive to acquire, several approaches [10, 11, 12, 13] very successfully incorporate text-only data to greatly improve quality.

Do we still need pronunciation lexica? On the one hand, having a E2E model seems to be a better strategy for accurate ASR because it directly optimizes for WER. However without paired text-audio data, it is unlikely that an E2E model would be able to recognize words that have unusual pronunciations given their spelling. Indeed, even native speakers who have never heard words like *Pfafftown* /p O: f . t aU n/ (first 'f' is silent), would have a hard time predicting their pronunciations. Given that the distribution of words in a language typically follows a Zipf distribution, adding more and more training examples yields diminishing improvements and pure text data would be unlikely to be helpful. An alternative for improving rare word performance was presented in [14], which showed that context biasing can improve recognition for rare words when a context can be present. For example, since a user's contacts are stored on the phone, they can be used as additional context to help recognition performance.

In this paper, we explore using a pronunciation lexicon in an E2E model, while still retaining the joint optimization benefit of E2E models by predicting graphemes. Specifically, past work [14] has focused on injecting context[1] into the E2E model in the form of graphemes, through a model known as contextualized Listen, Attend, and Spell (CLAS). We extend CLAS by injecting pronunciations into the context module, but still predict graphemes as output units from the model. From the outside, the model still behaves like an E2E model that takes audio and predicts text; there is no need for constructing or decoding with an optimized decoder graph, or the need to add a separate text normalization module. Our approach, which we refer to as Phoebe,[2] is able to inject additional phonetic knowledge while improving WER by up to 16% relative.

[1] By "*context*", we mean additional inputs beyond the audio that might be relevant to the recognition process (e.g., contact list, user's library of song names, etc.). Such information could be used to "*bias*" the recognition towards these words or phrases.

[2] The name comes from the authors misunderstanding each other when discussing "PHOnEtic Biasing for End-to-end models".

2. PRIOR WORK

2.1. Listen, Attend, and Spell (LAS)

The LAS model is an E2E ASR system based on the encoder-decoder architecture [4]. The light blue boxes in figure 1 correspond to this model.

The LAS model outputs a probability distribution $P(y|x)$ over sequences of output labels y (graphemes, in this work) conditioned on a sequence of input audio frames $x = (x_1, \ldots, x_U)$ (log-mel features, in this work). The model is trained to minimize $J^{LAS} = -\log P(y|x)$.

The encoder is composed of a stacked layer of unidirectional LSTMs, which embed the input sequence x into a sequence h^x. The decoder consumes these embedding via an attention mechanism that summarizes the inputs in a context vector c_t^x. The attention [15, 16] uses two inputs. The first input h^x is an embedding of the audio frames (the superscript x denotes the fact that it comes from audio). The second input is the output of the decoder at the previous step: d_{t-1}^x. The decoder, which is also composed from layers of unidirectional LSTMs, outputs d_t^x (first layer) and a vector y_t (last layer) for every decoding step t conditioned on the context vector c_t^x and on previous output y_{t-1}, and stops when an end-of-sequence token is emitted. We did not use scheduled sampling [17] for training.

2.2. Contextual LAS model (CLAS)

The Contextual LAS model (CLAS) is a modification of LAS, which allows it to take as input a set of bias phrases *in addition to* the usual sequence of audio frames. A biasing phrase could be "John Smith", an entry in the user's contact list. This model was shown to outperform other contextualization techniques in on some tasks [14] and is depicted in figure 1.

Each of the bias phrases $g_1 \ldots g_N$ is provided to CLAS as a sequence of graphemes (represented as one-hot encoding), which are then embedded into a set of fixed length vectors $h^g = (h_1^g \ldots h_N^g)$ using an LSTM. The CLAS decoder then accepts in each decoding step t both an *audio* vector c_t^x as well as a *biasing* vector c_t^g, obtained from attention computed over h^g (using the decoder's state d_{t-1}^g as attention-query).

Equation 1 describes the bias attention computation of a single bias phrase. It takes as inputs the bias grapheme embeddings h_i^g and the previous decoding state d_{t-1}^g and compute an attention weight - a set of weights u_{it}^g for decoding step t. The output is then normalized with a softmax to obtain α_t^g.

$$u_{it}^g = v^{gT} \tanh\left(W_h^g.h_i^g + W_d^g.d_t^g + b^g\right) \quad (1)$$

$$\alpha_t^g = \text{softmax}\left(u_t^g\right) \quad (2)$$

Then, the bias context vector is computed as a weighted sum of the embeddings h_i^g (Equation 3).

$$c_t^g = \sum_{i=1}^{N} \alpha_{it}^g h_i^g \quad (3)$$

One of the shortcomings of CLAS, as presented in [14], is that only grapheme embeddings are fed to the context module. Error analysis shows that E2E models struggle with rare words [9] especially those that have unusual pronunciations. In the next section, we address this shortcoming of CLAS by injecting pronunciations into the context module.

3. PHOEBE

3.1. Model

The new model is depicted in figure 2. Our objective with Phoebe is to improve problems with rare words by injecting lexical entries, without having to resort to building a model that outputs phonemes, which has its own pitfalls [9].

First, we keep the current output of the LAS architecture, and thus $P(y_t|y_{t-1} \ldots y_1; x)$ corresponds to a probability of graphemes only. Next, we need to specify the inputs to the decoder. Specifically, we decided that the decoder input c_t^g should not directly incorporate phonetic information and thus should only be a weighted sum of grapheme embeddings. Since the decoder operates on graphemes only, it seems natural to have an additional input in terms of graphemes only.

In order to add pronunciation information, we added a separate encoder, shown in green in figure 2. For each of the N biasing phrases, we look up the pronunciations and we obtain N strings of phonemes. The lookup is done with our current lexicon dictionary backed by a G2P for out-of-vocabulary words. Since we also use a no-biasing option like in [14], we have a corresponding empty string of phonemes for it. Each of the N strings of phonemes is encoded in a similar way as the biasing phrases themselves are: the phonemes are one-hot encoded and then go through an LSTM network of identical dimension as the one for the graphemes and we obtain h_p. The embedding is done independently and thus there is not necessarily any relationship between the embeddings of the graphemes and the phonemes.

Given both phoneme and grapheme embeddings (h^g and h^p), the next design choice was how to modify the bias attention. We can interpret attention as having two aspects. The first one is to compute a set of weights using equations 1 and 2. The second aspect is to use the set of weights to compute a weighted sum of embeddings using equation 3. While we want the attention to take advantage of both grapheme and phoneme information, we want c_t^g to be only a weighted sum of graphemes. Therefore, we decided to use both graphemes and phonemes to compute attention weights but once the attention weights were computed, we used them to compute a weighted sum of only the grapheme embeddings. With this approach, there is no a-priori relationship between spelling and pronunciation, and we can assign an arbitrary sequence of phonemes to any word.

Mathematically, this means that equations 2 and 3 above are left unchanged. However, we do modify equation 1 to also use the phoneme embeddings. The new terms are highlighted below where the square brackets represent a vector concatenation and P is a projection matrix:

$$u_{it}^g = v^{gT} \tanh\left(W_h^g.h_i^g + \boldsymbol{P.W_d^{gp}.[d_t^g; d_t^p]} + b^g\right) \quad (4)$$

By having both inputs, we guarantee that the new model is a strict superset of the old model. In the subsequent experiments, we kept the dimensions of the modules that are common between figures 1 and 2 identical (through the projection matrix), so that the only changes were due to the injection of pronunciations. As an alternative to concatenating phoneme and grapheme embeddings, pilot experiments considered using phoneme embeddings (d_t^p) alone but were not as successful.

3.2. Training

The sampling of which words to use as biasing inputs was similar to [14]. The training examples are used in batch, and for each

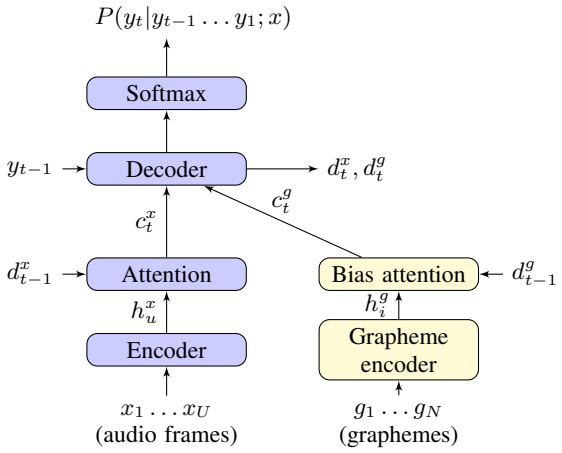

Fig. 1. The CLAS model. Light blue boxes correspond to the LAS blocks, and light yellow boxes to its contextual extension. Note that this model accepts bias phrases only in the form of graphemes.

batch, we randomly kept the reference transcript with probability P_{keep}. This allows to have examples for which there was no matching biasing. For the remaining reference transcripts, we randomly selected n-grams (where n is uniformly sampled from $[1, N_{\text{order}}]$). The number of selected n-grams was N (from figure 2) where N was uniformly sampled from $[1, N_{\text{phrases}}]$. In all cases, we also added a "no-biasing" option, where the biasing graphemes and phonemes were empty. This sampling approach ensured that some examples did not have matching words, while others did.

As in [14], we also introduced a special token to help the model converge. After each matching bias phrase, we added a special token `</bias>`. For example, if the training example is `average panda weight` and the biasing phrase is `panda`, then the transcript is modified to be `average panda</bias> weight`. It introduces a biasing error that can only be corrected by using the biasing phrase, as suggested by [18].

For the phonemes, we used a mix of lexicon and G2P [19]. If a word was in our lexicon, we used its corresponding pronunciation. In case there were multiple possible pronunciations, we randomly picked one of them. In case the word was not present in the lexicon, we used a G2P to predict the pronunciation and if it failed we used an empty string of phonemes. While ideally, we would know the correct pronunciation for every word in the biasing set, this would have required to have humans annotate our training data phonetically, and this was not available to us.

3.3. Inference

During the inference, we optionally provided contextual information. For example, for test sets where users wanted to contact someone (e.g. "*send a text message to Jane Doe*"), we created a bias phrase list that contained the correct contact name ("*Jane Doe*") and other non-matching names as distractors, with pronunciations obtained either from the lexicon of G2P as in training. We also ran experiments where the context information was always empty, to check whether phoneme injection impacts performance when these are not provided.

4. EXPERIMENTS

4.1. Experimental Setup

Our setup was similar to [15]. For training, we used about 35 million US English audio utterances ($\sim$ 27,500 hours). The training utterances are anonymized and hand-transcribed, and are representative of Google's voice search traffic. In addition, we added noise and reverberation during training as in [20].

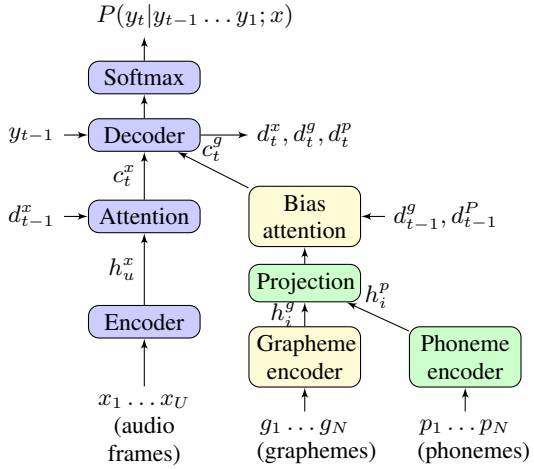

Fig. 2. The Phoebe Model. Here the light green boxes correspond to the phonetic components added to the CLAS model on the left.

	Test set	CLAS	Phoebe	Change
`E1`	Voice search	6.3	6.3	nil
`E2`	Dictation	5.6	6.1	+9%

Table 1. WER on CLAS (Figure 1) and Phoebe (Figure 2) models.

The models evaluated in this section are trained on 8×8 Tensor Processing Units (TPU) [21] slices with global batch size of 4,096. Each training core operates on a shard-size of 32 utterances in each training step. From this shard, bias phrases are randomized and thus each shard receives a maximum of 32 bias phrases during training.

The LAS model architecture is as follows. The encoder has 10 LSTM layers, each having 256 nodes and a hidden size of 1,400. Multi-head attention with 4 heads is used with context vectors of dimension 512. Finally, the decoder consisted of 4 LSTM layers, each of size 256 with a hidden dimension of 1,024. The grapheme and phoneme encoders had the same setup. They both were single layer LSTMs with 512 dimensions.

As in [14], we used $P_{\text{keep}} = 0.5$, $N_{\text{phrases}} = 1$, and $N_{\text{order}} = 4$.

4.2. Results

As a check, we first measured the performance of both models on sets of general traffic, shown in Table 1. The first set was a random sampling of queries to perform web searches (`E1`). The second set was a random sampling of dictation queries such as for the content SMS or email messages (`E2`). In both experiments, the biasing list was empty. For both CLAS and Phoebe, we had picked the best training step based on performance of the set of `E1` (there was minimal variability) and we see that the performance is roughly identical. For the dictation set `E2`, there is a degradation of about 9%. In practice, we feel that it is not a major issue since we already use distinct models.

Next, we measured the quality of our new model on a test set that consisted of users issue voice commands to make a phone call to a contact (e.g. "*Call John Smith's cell phone*"). The test set was human-transcribed to contain about 5,000 utterances sampled from real traffic. For each of the utterances, we had a biasing context of about 200 names, with one of them being the correct name, and the others being distractors. The results are shown on table 2 (`E3`). We see a relative improvement of word-error-rate (WER) of about

	Test set	CLAS	Phoebe	Change
E3	Real	12.2	10.2	-16%
E5	[no bias]	[17.2]	[17.9]	
E4	TTS	11.8	10.5	-11%
E6	[no bias]	[28.7]	[28.4]	

Table 2. WER for CLAS (Figure 1) and Phoebe (Figure 2) models for contact utterances.

	Word type	Count	CLAS	Phoebe	Change
E3	Common	12,184	9.1	7.7	-15%
E3	Rare	1,396	26.9	20.6	-23%

Table 3. WER on contacts set with biasing (E3 from table 2) segregated by word frequency.

16% from 12.2 to 10.2. For reference, we also include in square brackets the WER for when the biasing context is removed (E5). In practice, the context is present, so these higher WER numbers are not experienced by the users.

4.3. Analysis

To confirm our intuition that the improvements were due to the phonetic injection, we measured the performance on common and rare words. We segregated words by their frequency in a large text corpus. Depending on whether the word was among the 32,000 most common ones or not, we declared it "common" or "rare". We then classified the words of E3 from table 2 and measured their WER (Table 3). Unsurprisingly, the WER for common words was lower than the average for both CLAS (9.1 vs 12.2) and Phoebe (7.7 vs 10.2). For uncommon words, the situation was flipped for both CLAS (26.9 vs 12.2) and Phoebe (20.6 vs 10.2). We also confirmed that the relative improvements were greater for rare words (-23% relative) than for common words (-15% relative).

Finally, we wanted to observe the performance on individual transcripts. To do this, we synthesized a set that does not use real users' data. Using a sample of names, we used Google's text-to-speech (TTS) system WaveNet [22] to synthesize audio. We then augmented the context with additional unrelated names, and ran recognition. We report the overall performance of the models on table 2 (E4 and E6). These numbers are not true measures of the performance of the models, since we used the same lexicon for both ASR and TTS, thus making the problem easier. Nonetheless, the relative changes mostly agree with E3.

We report examples that we felt were typical of the behavior of the new models in Table 4. We see in examples A and B that the improvements come from unusual names, even if in example D the improvements are not always present. In addition, the new model seem to be making slightly more mistakes with words that are not part of the name, as show in example C.

	Reference	CLAS	Phoebe
A	call nicola mondesir from contacts	call nicola mondesier from contacts	call nicola mondesir from contacts
B	call knaub vickie work	call kanab vicki work	call knaub vickie work
C	video call kosek	video call kosek	video called kosek
D	hey google can i call hughley	hey google can i call hughley	hey google can i call hughle
E	call my Gilda at work	oh i know that works	call my gilda at work

Table 4. Example of individual results on TTS set (E4) with correct words in blue and incorrect ones in red.

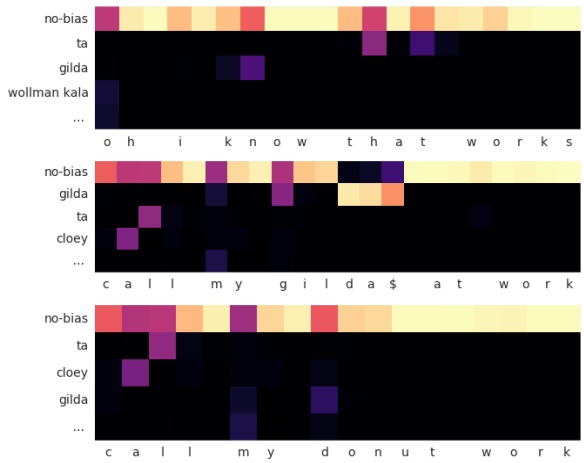

Fig. 3. Attention weights α_t^g from equation 2 for row E of table 4 for CLAS (top), Phoebe (middle), and Phoebe with incorrect pronunciation (bottom).

Finally, we sought out an extreme example of where Phoebe outperformed CLAS, shown on row E of table 4. On figure 3, we plotted the bias attention weights for the biasing phrases (y-axis) as a function of the decoding step (x-axis). The correct biasing phrase is "gilda" and the other words such as "cloey" are distractors. We see that the Phoebe model is able to attend to the correct name, thus resulting in an improvement of the recognition. If we *purposely* degrade the system by changing the pronunciation for the word "gilda" to be /r\ oU z @ l i/ ("rosalie"), we see that the model no longer attend to it and its predicted transcript is "call my donut work", thus demonstrating that the phonetic information is being used.

5. CONCLUSION

In this paper, we presented an improvement on biasing for E2E ASR models. By injecting pronunciation knowledge into an extension of the model of [14], we were able to get a 16% relative improvement over the baseline model. The new model still retains the advantages of E2E models (simple and unified training, implicit learning of pronunciation of common words), while at the same time incorporates knowledge of tail words' pronunciation even if they have never been seen during training.

We have thus taken a step towards addressing the challenges presented in [9] for biasing. Future work will focus on injecting knowledge of unseen words when biasing information is not readily available.

6. REFERENCES

[1] Hervé Bourlard and Nelson Morgan, *Connectionist speech recognition*, Kluwer Academic Publishers, 1994.

[2] R Prabhavalkar, K Rao, TN Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly, "A comparison of sequence-to-sequence models for speech recognition," *Interspeech*, 2017.

[3] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," *ASRU*, 2017.

[4] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals, "Listen, attend and spell," *ICASSP*, 2016.

[5] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philémon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," *ICASSP*, 2016.

[6] Shinji Watanabe, Takaaki Hori, Jonathan Le Roux, and John R. Hershey, "Student-teacher network learning with enhanced features," *ICASSP*, 2017.

[7] Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny, "Building competitive direct acoustics-to-word models for english conversational speech recognition," *ICASSP*, 2018.

[8] Amit Das, Jinyu Li, Rui Zhao, and Yifan Gong, "Advancing connectionist temporal classification with attention modeling," *ICASSP*, 2018.

[9] Tara N. Sainath, Rohit Prabhavalkar, Shankar Kumar, Seungji Lee, Anjuli Kannan, David Rybach, Vlad Schogol, Patrick Nguyen, Bo Li, Yonghui Wu, Zhifeng Chen, and Chung-Cheng Chiu, "No need for a lexicon? evaluating the value of the pronunciation lexica in end-to-end models," *ICASSP*, 2018.

[10] Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates, "Cold fusion: Training seq2seq models together with language models," *Interspeech*, 2018.

[11] Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N. Sainath, Zhifeng Chen, and Rohit Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," *ICASSP*, 2018.

[12] Adithya Renduchintala, Shuoyang Ding, Matthew Wiesner, and Shinji Watanabe, "Multi-modal data augmentation for end-to-end asr," *Interspeech*, 2018.

[13] Shigeki Karita, Shinji Watanabe, Tomoharu Iwata, Atsunori Ogawa, and Marc Delcroix, "Semi-supervised end-to-end speech recognition," *Interspeech*, 2017.

[14] Golan Pundak, Tara N. Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao, "Deep context: End-to-end contextual speech recognition," *SLT*, 2018.

[15] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," *ICASSP*, 2018.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *NIPS*, 2017.

[17] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *NIPS*, 2015.

[18] Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, and Ian McGraw, "Streaming small-footprint keyword spotting using sequence-to-sequence models," *ASRU*, 2017.

[19] Martin Jansche, "Computer-aided quality assurance of an icelandic pronunciation dictionary," in *LREC*, 2014.

[20] Chanwoo Kim, Ananya Misra, Kean Chin, Thad Hughes, Arun Narayanan, Tara Sainath, and Michiel Bacchiani, "Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," *Interspeech*, 2017.

[21] Norman P. Jouppi et al, "In-datacenter performance analysis of a tensor processing unit," *SIGARCH Comput. Archit. News*, 2017.

[22] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, 2016.