INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Improving Performance of End-to-End ASR on Numeric Sequences

Cal Peyser, Hao Zhang*, Tara N. Sainath, Zelin Wu*

Google Inc., U.S.A

{cpeyser,haozhang,tsainath,zelinwu}@google.com

Abstract

Recognizing written domain numeric utterances (e.g., I need $1.25.) can be challenging for ASR systems, particularly when numeric sequences are not seen during training. This out-of-vocabulary (OOV) issue is addressed in conventional ASR systems by training part of the model on spoken domain utterances (e.g., I need one dollar and twenty five cents.), for which numeric sequences are composed of in-vocabulary numbers, and then using an FST verbalizer to denormalize the result. Unfortunately, conventional ASR models are not suitable for the low memory setting of on-device speech recognition. E2E models such as RNN-T are attractive for on-device ASR, as they fold the AM, PM and LM of a conventional model into one neural network. However, in the on-device setting the large memory footprint of an FST denormer makes spoken domain training more difficult. In this paper, we investigate techniques to improve E2E model performance on numeric data. We find that using a text-to-speech system to generate additional numeric training data, as well as using a small-footprint neural network to perform spoken-to-written domain denorming, yields improvement in several numeric classes. In the case of the longest numeric sequences, we see reduction of WER by up to a factor of 8.

1. Introduction

An ongoing challenge of ASR systems is to model transcriptions that do not exactly reflect the words spoken in an utterance. For example, the spoken utterance "set an alarm for four fifteen" is typically decoded in the written form as "set an alarm for 4:15". Numeric utterances, such as addresses, phone numbers, and postal codes are particularly hard members of this category, due to the inherent out-of-vocabulary (OOV) issues of long written-domain numeric sequences. This problem arises because of a data sparsity issue, namely that long numeric sequences are unlikely to be present in training data.

A natural solution to the data sparsity/OOV problem is to train an ASR system in the spoken domain, and then to denorm the results back into the written domain [1, 2, 3]. Such a solution for conventional ASR systems was presented in [4]. In that work, the AM and PM are trained in the spoken domain. A verbalizer weighted finite state transducer (WFST) is then inserted before the LM to convert commonly used spoken-domain numerics to the written domain. Finally, a class-based n-gram LM, modeled as an FST, is used to transduce additional spoken domain entities to the written domain.

There are a few issues with the above approach for handling the numerics problem. First, it is based on a set of rules in the verbalizer and class-based LM, which does not scale well to changes in training data [5]. In addition, the production FST-LM is large, which presents a challenge for memory-constrained on-device applications [6].

End-to-end (E2E) ASR models have gained popularity in recent years. These models, which fold the AM, PM and LMs into a single neural network, are particularly attractive for on-device applications. E2E models, including RNN-T, have been shown to achieve comparable performance to conventional models at a fraction of the size [7, 6]. However, these same constraints preclude the use of an FST denormer, and thus make spoken domain training more difficult.

The goal of this work is to improve E2E performance on numerics by applying several techniques. Synthetic data has been used successfully to address data sparsity issues in image processing for scene text recognition [8], text spotting [9], object detection [10] and speech recognition [11]. Given this success, we first explore augmenting our training set using a text-to-speech (TTS) system to synthesize additional written-domain numeric training data. In doing so, we choose transcripts from several challenging numerics categories in order to improve coverage of those cases.

Data-sparsity/OOV issues have also been addressed by neural correction models in both in NLP [12, 13] and in speech [14, 15, 16, 17, 18]. A recent example in ASR is [17], which proposed a neural error corrective language model (NECLM) trained on pairs of transcripts and corresponding errorful ASR output as a last step during decoding. This approach was applied to denorming in [18] with a neural written-to-spoken denormalizer. We build on the work in [18] by training a "neural correction" network, which is trained on written-domain ground-truth/RNN-T hypothesis pairs and learns to correct mistakes.

Given the success of the spoken domain for numeric entities, a third avenue we explore is to train RNN-T in the spoken domain and denorm back to written domain. We compare both a traditional FST-based denormer as in [4] and a "neural denormer" that is based on our written correction model.

We evaluate the different E2E numeric solutions on a variety of different numeric test sets, which differ in their numeric sequence length. We find that while written domain approaches show significant improvements, the largest improvements are in the spoken domain, in which the OOV and data-sparsity issues are mitigated, and particularly for long numeric sequences. Specifically, in the written domain, the addition of TTS training data gives a 65% relative WER improvement for small numbers and a 35% improvement for large numbers, increasing to 75% and 49% with the addition of a neural correction model. In the spoken domain, we see our best results, where the incorporation of a neural denorming model yields a 75% improvement on short numbers and 83% on large numbers. On a curated eval set of specifically difficult verbalizations, results on long utterances are even stronger, showing a factor of 8 reduction in WER.

The rest of this paper is organized as follows. Section 2 introduces the four variations of standard RNN-T that we worked with. This section includes our procedure for TTS utterance generation as well as our neural correction/denorm architecture. Section 3 describes the details of our experiments. Section 4 presents our results and analysis. Finally, Section 5 summarizes our conclusions.

*Contributed equally.

http://dx.doi.org/10.21437/Interspeech.2019-1345

Table 1: *Sample TTS Utterance Categories, with Example Synthetic Numerics*

Category	Example Transcript Template	Example Numeric	Average Numeric Length
DAY	remind me on monday the $DAY	31st	1.8
PERCENT	turn down sound to $PERCENT	20.22%	2.2
POSTALCODE	how far away is $POSTALCODE	86952	5.1
TIME	set second alarm for $TIME p.m.	10:46	3.0
YEAR	play the top 40 from $YEAR	1648	4.0

2. Methods

In this section, we present different ideas explored to address the long-tail numeric issue of our RNN-T system. We give each approach a label, which we will reference in Table 2 below.

2.1. TTS Training Data (`W1`)

To address the numeric data-sparsity issue, we generate additional training data that represent challenging and realistic numeric sequences. To this end, we choose numeric categories that we see often in Google Assistant traffic. See Table 1 for sample categories. We specifically choose categories that represent a variety of numeric sizes. To obtain transcripts in these categories, we choose 200 unique templates for each category from anonymized Google Assistant utterances, and inject the numeric entities by performing weighted sampling from our production numeric WFST grammar (weighted on the spoken domain). Two parallel transcripts are generated for each template, one in the spoken domain and the other in the written domain. We perform this process 100 times for each template, each time synthesizing a unique utterance. All models below use TTS training data generated in this manner.

2.2. Written-Domain Neural Correction (`W2`)

We seek to correct written RNN-T hypotheses with a post-processing neural correction step. Our written-domain correction model, shown in Figure 1, is an attention-based sequence-to-sequence model, for which the input is RNN-T hypotheses and the output is a corrected transcript. Our architecture is adapted to the correction setting by accounting for the fact that the majority of words in an input sentence are simply copied to the output during correction. This is done by training an additional "tagger" RNN to run on the input sequence before the sequence-to-sequence model. The tagger tags each word in the input sequence as either "trivial", in which case it is simply copied to the output sequence, or "non-trivial", in which case it is passed into the attention model and decoder. This architecture has been reported for contextual text normalization for text-to-speech [5], but we adapt it here to the correction setting for ASR. While we use the model for correction, it could also be used to re-rank an n-best list in a second-pass setting.

2.2.1. Encoder Layer

Suppose we seek to map between input sequence $\mathbf{x} = \{x_1, \ldots, x_I\}$ and output sequence $\mathbf{y} = \{y_1, \ldots, y_T\}$, where the sequence vocabulary is composed of words. We define a BiRNN encoder such that

$$h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$$

where

$$\overrightarrow{h}_i = \mathbf{RNN}_{\rightarrow}(\overrightarrow{h}_{i-1}, x_i)$$

and

$$\overleftarrow{h}_i = \mathbf{RNN}_{\leftarrow}(\overleftarrow{h}_{i+1}, x_i)$$

where $\mathbf{h} = h_1, \ldots, h_I$ are hidden encoder states.

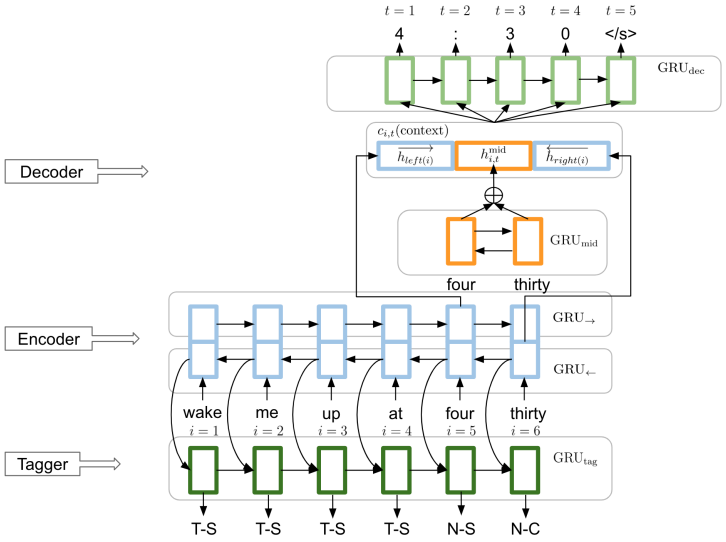

Figure 1: *Neural Denormer Architecture. T stands for* trivial. *N stands for* non-trivial. *S stands for* start. *C stands for* continuation.

2.2.2. Tagging Layer

We define the "tagger" RNN as

$$s_i = \mathbf{RNN}_{\text{tag}}(s_{i-1}, t_{i-1}, h_i)$$

where $\mathbf{s} = s_i, \ldots, s_I$ are hidden tagger states, with corresponding observations, i.e., tag sequence $\mathbf{t} = t_i, \ldots, t_I$.

Each tag t_i is a joined tag in the cross-product set of $\{\text{trivial}, \text{non-trivial}\} \times \{\text{start}, \text{continuation}\}$ to model whether a word is the beginning of a new segment needed to be corrected or a continuation of the previous segment. This refinement allows us to model consecutive non-trivial segments.

The learning objective of the tagger is

$$\underset{\mathbf{t}}{\text{argmax}} \prod_{i=1}^{I} P(t_i|s_i) \quad (1)$$

where P is defined as a linear projection of $\mathbf{s}$ followed by a softmax layer. We obtain alignments for training the tagger by using a heuristic alignment algorithm that determines subsequences common to the input and output transcripts. These common subsequences are marked as "trivial".

2.2.3. Decoder Layer

We use the results of the tagger to extract text snippets to be corrected. Suppose a text snippet spans from time s to e. The input span $\{x_s, \ldots, x_e\}$ along with the context hidden states $\overrightarrow{h}_s$ and $\overleftarrow{h}_e$ become the input to the next stage attentional model. We define a BiRNN encoder $\mathbf{RNN}_{\text{mid}}$ over $\{x_s, \ldots, x_e\}$.

Finally, the attentional RNN decoder $\mathbf{RNN}_{\text{dec}}$ is defined as

$$d_{i,t} = \mathbf{RNN}_{\text{dec}}(d_{i,t-1}, y_{i,t-1}, c_{i,t})$$

where $c_{i,t}$ is the result of the attention function of $d_{i,t-1}$, $\overrightarrow{h}_s$, $\overleftarrow{h}_e$, and $\mathbf{RNN}_{\text{mid}}(\{x_s, \ldots, x_e\})$. The two-dimentional indices (i, t) indicate t is relative to a given position i (s, e) in the input sentence. The architecture of the attention component is the same as in [5].

The learning objective of the decoder is

$$\underset{\mathbf{y}}{\text{argmax}} \prod_{t=1}^{L(i)} P(y_{i,t}|d_{i,t}) \quad (2)$$

In this way, the attention mechanism and decoder are applied only to relevant spans of text, which decreases cost and improves accuracy. As shown in the derivation steps, at training time the two objectives in Equation 1 and Equation 2 translate to two cross-entropy losses that can be linearly combined during training. At decoding time, the two models work as a pipeline, with the attentional model only used as required by the tagging model.

2.3. Spoken Domain Training, FST Denorming (S1)

Following the success in conventional ASR models, we explore spoken domain training for RNN-T. We train RNN-T on a spoken-domain version of our training set, and leave the translation back to written domain to an FST denormer derived from our production grammar [4].

In order to train this model, we require examples of utterance transcripts in both the spoken and written domains. We gather these examples by passing written-domain transcripts from our training set through an FST verbalizer. We then choose a single verbalization by passing each candidate hypothesis through a lexicon, and force-aligning the resulting phone sequences against the phones in the utterance. For TTS training data, we simply use the spoken domain transcript that we obtained using our verbalization grammar.

2.4. Neural Denorming (S2)

Since the FST-based denorming approach discussed in Section 2.3 is challenging to put on device, we also explore a neural denorming approach. Specifically, we adapt our written correction model from Section 2.2 to the spoken domain by rephrasing it as a denormer which consumes spoken domain training data and emits written domain output. The architecture of the neural denorming model is identical to the written correction model.

3. Experiments

3.1. Data Sets

Our experiments are conducted on a ∼30,000 hour training set consisting of 43 million English utterances. The training utterances are anonymized and hand-transcribed, and are representative of Google's voice search traffic in the United States. Multi-style training (MTR) data are created by artificially corrupting the clean utterances using a room simulator, adding varying degrees of noise and reverberation with an average SNR of 12dB [19]. The noise sources are drawn from YouTube and daily life noisy environmental recordings. The real audio test sets we report results on include ∼14.8K voice search (VS) utterances extracted from Google traffic, and real-audio numeric test set of ∼8.0K utterances derived from the VS distribution (NUMERICS).

In order to generate synthetic training data, we use a multi-speaker TTS system based on the baseline architecture as described in [20], where the Tacotron model [21] generates a mel-spectrogram conditioned on phonemes and a 64-dimensional speaker embedding learned for each speaker during training. The predicted mel-spectrogram is then inverted to time-domain waveform with a WaveRNN neural vocoder [22]. We again use MTR to add artificial noise to our synthesized audio. The TTS training data consists of 84 English speakers covering American, Australian, British and Singaporean accents with a Google Assistant clean speaking style, comprising 370 hours of audio in total. During inference, numeric input texts are mapped to phonemes and a speaker is randomly selected.

This system is used to synthesize audio for templated transcripts, as described above. In this manner we create training sets for each numeric category, and create our final training set by grouping all categories together. For each batch in training, we draw 10% of training examples from this TTS set and 90% of examples from the real-audio training set described above.

Since we know that the length of a numeric sequence exacerbates the OOV issue, for our test sets we split numerics into three categories based on length. For these categories, we generate transcripts using the same template phrases as in training, but with unique verbalizations, to ensure that no phrase is used in both training and test. We call these sets the SAMPLED_SHORT, SAMPLED_MEDIUM and SAMPLED_LONG test sets. We are also interested in evaluating our models on the "tail" of the distribution - that is, on the most difficult numerics. To this end, for each category we maintain a list of particularly difficult numeric verbalizations. For example, the postal code "22110" might be verbalized as "double two double one oh". We call the test sets generated with these more difficult verbalizations TAIL_SHORT, TAIL_MEDIUM, and TAIL_LONG.

In order to ensure that our system has not overfit to a specific sort of TTS data, we synthesize audio for our test sets using separate TTS systems. The audio for the SAMPLED sets is created using a single-speaker WaveNet [23] system. The audio for the TAIL sets is created using a concatenative TTS [24] system.

3.2. RNN-T

All experiments use the same RNN-T architecture, following the setup in [6]. Specifically, an 8-layer unidirectional LSTM of width 2,048 with 640 recurrent projection units is used as the encoder. A time-reduction layer with the reduction factor 2 is inserted after the second LSTM layer of encoder to reduce model latency. The RNN-T decoder consists of a prediction network with 2 LSTM layers of 2,048 hidden units and a 640-dimensional projection per layer as well as an embedding layer of 128 units. The outputs of the encoder and prediction network are fed to a joint network that has 640 hidden units, followed by a 4,096 wordpiece softmax output. In total, RNN-T has approximately 114M parameters. The RNN-T model is implemented in Tensorflow [25] and trained on 8×8 Tensor Processing Units (TPU) slices with a global batch size of 4,096 for $\sim 200K$ steps.

3.3. Neural Correction/Denorming

The neural denormer uses a bidirectional single-layer GRU encoder with 256 units that emits a 256-dimensional hidden state. The tagging RNN is a single-layer GRU with 64 units. The decoder is a bidirectional single-layer GRU with 256 units. The encoder/tagger portion of the model, which runs for all input, contains about 2M parameters, while the attention/decoder portion of the model, which runs only for text spans marked for correction, contains about 4M parameters. The small footprint of the neural correction model makes it attractive for the on-device setting. The model is implemented in Tensorflow and trained asynchronously on 12 GPUs, with a batch size of 16.

Table 2: *Results: WER by Model and Test Set*

		Written	Written TTS	Written TTS Neural Correction	Spoken TTS FST Denorm	Spoken TTS Neural Denorm
	Average Num Len	W0	W1	W2	S1	S2
VS		6.7	6.8	6.5	7.6	6.9
NUMERICS	3.0	6.9	6.6	6.3	8.3	7.2
SAMPLED_SHORT	2.1	7.6	2.6	2.1	4.2	1.9
SAMPLED_MEDIUM	3.3	11.9	5.7	5.2	7.6	4.0
SAMPLED_LONG	7.4	26	16.9	13.2	16.8	4.4
TAIL_SHORT	2.7	6.8	4.9	4.7	7.3	4.2
TAIL_MEDIUM	4.3	15.4	12.9	11.6	11.6	6.7
TAIL_LONG	6.9	92.7	74.8	50.9	11.9	11.3

4. Results

Table 2 gives WER results for each of our experiments on the SAMPLED and TAIL test sets, as well as the real-audio VS and NUMERICS test sets. We use the labels given in Section 2 (W1, W2, S1, S2) for convenience. We use W0 to refer to the baseline RNN-T model.

The results for the written domain models are characterized by a steep decline in quality with increasing numeric length. While the the addition of TTS data in W1 and the incorporation of a correction model in W2 improve over the baseline W0, the error rates are still sometimes higher on the MEDIUM and LONG test sets than on the non-numeric VS set.

The errors made by model tell a clear story: the written domain models struggle to correctly format numerics, and often confuse parts of numeric sequences for similar-sounding words. The following are representative errors taken from the SAMPLED sets. Here, references are on the left of the arrow and erroneous hypothesis are on the right.

W1:

"$180.50 into inr" → "**$180 - $50** in inr"

"house for rent 60003" → "house for rent **$6003**"

W2:

"$487 / 6" → "**4876**"

"48007 to 08618" → "**480-708-6618**"

In the spoken domain, we see a different error pattern. S1 errors show that the FST denormer often completely or partially fails to perform denorming, for example:

S1:

"code 30441" → "code **300 double 41**"

"the 32nd door" → "the **30 second door**"

This indicates the problem with rule-based FST denormers, which often cannot adapt well to changes in training data.

Progressing to S2, these issues largely disappear, and the errors that remain are a blend of formatting, denorming, and other errors, in considerably smaller numbers than for all other models. It seems that the avoidance of OOV issues obtained by training in the spoken domain largely solves the formatting problems experienced by the written domain models, while using a neural denormer, which learns how to denorm from data, sidesteps the denorming errors seen in the FST-based spoken domain model. Finally, the spoken denorming approach does not result in a significant degradation on the real-audio VS or NUMERIC sets.

5. Conclusions

In this paper, we experimented with four approaches for improving end-to-end ASR performance on numeric utterances. We found that all approaches yield improvements, with the largest improvements occurring when TTS training data, spoken domain training, and neural denorming are all used together. The fact that we see the largest improvements in the longest utterances, together with the types of errors that seem to be avoided, suggest that the improvements are mainly due to the alleviation of the out of vocabulary problem.

6. Acknowledgements

We thank Gabriel Mechali, Mark Epstein, Michael Riley, and Richard Sproat for help and comments on this work.

7. References

[1] M. Shugrina, "Formatting time-aligned ASR transcripts for readability," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of he Association for Computational Linguistics*, ser. HLT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 198–206. [Online]. Available: http://dl.acm.org/citation.cfm?id=1857999.1858022

[2] H. Sak, C. Allauzen, K. Nakajima, and F. Beaufay, "Language model verbalization for automatic speech recognition," in *Proc. ICASSP*, 2013.

[3] C. Chelba, "Query Language Modeling for Voice Search," in *Proc. IEEE Workshop on Spoken Language Technology*, 2010.

[4] K. H. Lucy Vasserman, Vlad Schogol, "Sequence-based class tagging for robust transcription in asr," in *INTERSPEECH*, 2015.

[5] H. Zhang, R. Sproat, A. Ng, F. Stahlberg, X. Peng, K. Gorman, and B. Roark, "Neural models of text normalization for speech applications," *Computational Linguistics*, vol. 45, no. 2, 2019.

[6] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein, "Streaming End-to-end Speech Recognition For Mobile Devices," 2019.

[7] C. Chiu, 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 *Proc. ICASSP*, 2018.

[8] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," *Conference on Neural Information Processing Systems*, vol. abs/1406.2227, 2014. [Online]. Available: http://arxiv.org/abs/1406.2227

[9] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," *Computer Vision and Pattern Recognition*, vol. abs/1604.06646, 2016. [Online]. Available: http://arxiv.org/abs/1604.06646

[10] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," *Conference on Computer Vision and Pattern Recognition*, vol. abs/1804.06516, 2018. [Online]. Available: http://arxiv.org/abs/1804.06516

[11] Y. He, T. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S. yiin Chang, K. Rao, and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices," 2019. [Online]. Available: https://arxiv.org/abs/1811.06621

[12] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, ser. Conference on Neural Information Processing Systems. Cambridge, MA, USA: MIT Press, 2000, pp. 893–899. [Online]. Available: http://dl.acm.org/citation.cfm?id=3008751.3008881

[13] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," 09 2012.

[14] S. Kombrink, T. Mikolov, M. Karafiát, and L. Burget, "Recurrent neural network based language modeling in meeting recognition," in *INTERSPEECH*, 2011.

[15] O. Tilk and T. Alumäe, "Multi-domain recurrent neural network language model for medical speech recognition," 09 2014.

[16] J. Guo, T. N. Sainath, and R. J. Weiss, "A Spelling Correction Model for End-to-End Speech Recognition," 2019.

[17] T. Tanaka, R. Masumura, H. Masataki, and Y. Aono, "Neural error corrective language models for automatic speech recognition," in *Proc. Interspeech 2018*, 2018, pp. 401–405. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1430

[18] R. Sproat and N. Jaitly, "RNN approaches to text normalization: A challenge," *arXiv preprint*, vol. abs/1611.00068, 2016. [Online]. Available: http://arxiv.org/abs/1611.00068

[19] 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.

[20] W.-N. Hsu, Y. Zhang, R. Weiss, H. Zen, Y. Wu, Y. Wang, Y. Cao, Y. Jia, Z. Chen, J. Shen, P. Nguyen, and R. Pang, "Hierarchical generative modeling for controllable speech synthesis," in *Proc. ICLR, 2019, to appear*, 2019.

[21] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: A fully end-to-end text-to-speech synthesis model," *Proc. Interspeech*, 2017. [Online]. Available: http://arxiv.org/abs/1703.10135

[22] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," *ICML*, vol. abs/1802.08435, 2018. [Online]. Available: http://arxiv.org/abs/1802.08435

[23] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, "Parallel wavenet: Fast high-fidelity speech synthesis," *ICML*, vol. abs/1711.10433, 2018. [Online]. Available: http://arxiv.org/abs/1711.10433

[24] X. Gonzalvo, S. Tazari, C. an Chan, M. Becker, A. Gutkin, and H. Silen, "Recent advances in google real-time hmm-driven unit selection synthesizer," 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.