Fast, Compact, and High Quality LSTM-RNN Based Statistical Parametric Speech Synthesizers for Mobile Devices

Heiga Zen, Yannis Agiomyrgiannakis, Niels Egberts, Fergus Henderson, Przemysław Szczepaniak

Google

{heigazen,agios,nielse,fergus,pszczepaniak}@google.com

Abstract

Acoustic models based on long short-term memory recurrent neural network (LSTM-RNN) were applied to statistical parametric speech synthesis (SPSS) and showed significant improvements in naturalness and latency over those based on hidden Markov models (HMMs). This paper describes further optimizations of LSTM-RNN-based SPSS for deployment on mobile devices; weight quantization, multi-frame inference, and robust inference using an ϵ-contaminated Gaussian loss function. Experimental results in subjective listening tests show that these optimizations can make LSTM-RNN-based SPSS comparable to HMM-based SPSS in runtime speed while maintaining naturalness. Evaluations between LSTM-RNN-based SPSS and HMM-driven unit selection speech synthesis are also presented.

Index Terms: statistical parametric speech synthesis, recurrent neural networks.

1. Introduction

Statistical parametric speech synthesis (SPSS) [1] based on artificial neural networks (ANN) has became popular in the text-to-speech (TTS) research area in the last few years [2–20]. ANN-based acoustic models offer an efficient and distributed representation of complex dependencies between linguistic and acoustic features [21, 22] and have shown the potential to produce natural sounding synthesized speech [2, 4, 7–9]. Recurrent neural networks (RNNs) [23], especially long short-term memory (LSTM)-RNNs [24], provide an elegant way to model speech-like sequential data that embodies short- and long-term correlations. They were successfully applied to acoustic modeling for SPSS [8–11]. Zen *et al.* proposed a streaming speech synthesis architecture using unidirectional LSTM-RNNs with a recurrent output layer [9]. It enabled low-latency speech synthesis, which is essential in some applications. However, it was significantly slower than hidden Markov model (HMM)-based SPSS [25]. in terms of real-time ratio [26]. This paper describes further optimizations of LSTM-RNN-based SPSS for deployment on mobile devices. The optimizations conducted here include reducing computation and disk footprint, as well as making it robust to errors in training data.

The rest of this paper is organized as follows. Section 2 describes the proposed optimizations. Experiments and subjective evaluation-based findings are presented in Section 3. Concluding remarks are shown in the final section.

2. Optimizing LSTM-RNN-based SPSS

Figure 1 shows the overview of the streaming synthesis architecture using unidirectional LSTM-RNNs [9]. Unlike HMM-based SPSS, which usually requires utterance-level batch pro-

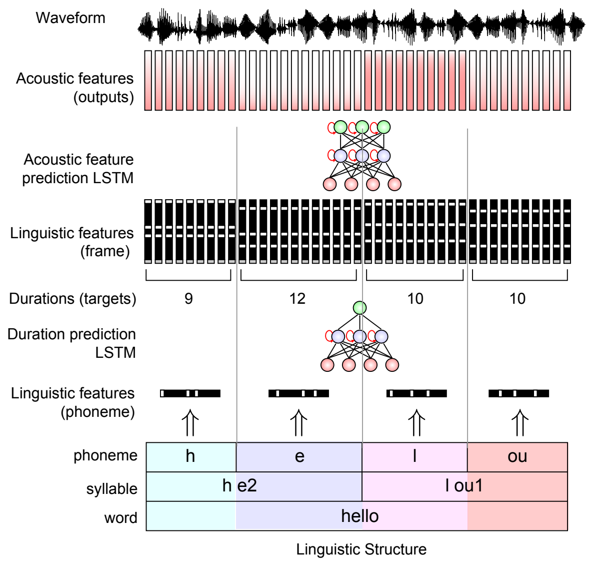

Figure 1: *Overview of the streaming SPSS architecture using LSTM-RNN-based acoustic and duration models [9].*

cessing [27] or frame lookahead [28], this architecture allows frame-synchronous streaming synthesis with no frame lookahead. Therefore this architecture provides a much lower latency speech synthesis. However, there are still a few drawbacks;

- **Disk footprint**; Although the total number of parameters in LSTM-RNN-based SPSS can be significantly lower than that of HMM-based SPSS [9], the overall disk footprint of the LSTM-RNN system can be similar or slightly larger because HMM parameters can be quantized using 8-bit integers [29]. Therefore decreasing the LSTM-RNN system disk footprint is essential for deployment on mobile devices.
- **Computation**; With HMM-based SPSS, inference of acoustic parameters involves traversing decision trees at each *HMM state* and running the speech parameter generation algorithm [27]. On the other hand, inference of acoustic parameters with LSTM-RNN-based SPSS involves many matrix-vector multiplications at each *frame*, which are expensive. This is particularly critical for client-side TTS on mobile devices, which have less powerful CPUs and limited battery capacity.
- **Robustness**; Typical ANN-based SPSS relies on fixed phoneme- or state-level alignments [2], whereas HMMs can be trained without fixed alignments using the Baum-

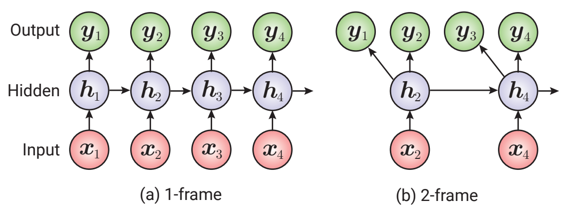

Figure 2: *Illustration of computation graph of (a) single-frame and (b) multi (two)-frame LSTM-RNNs.*

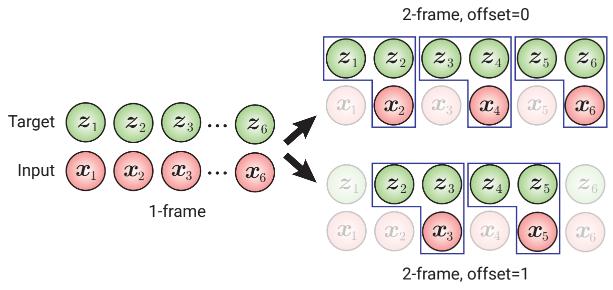

Figure 3: *Data augmentation with different offsets for 2-frame bundled inference.*

Welch algorithm. Therefore, the ANN-based approach is less robust to alignment errors.

This section describes optimizations addressing these drawbacks. Each of them that follow will be evaluated in Section 3.

2.1. Weight quantization

ANN weights are typically stored in 32-bit floating-point numbers. However there are significant advantages in memory, disk footprint and processing performance in representing them in lower integer precision. This is commonly approached by quantizing the ANN weights. This paper utilizes 8-bit quantization of ANN weights [30] to reduce the disk footprint of LSTM-RNN-based acoustic and duration models. Although it is possible to run inference in 8-bit integers with quantization-aware training [30], that possibility is not utilized here; instead weights are stored in 8-bit integer on disk then recovered to 32-bit floating-point numbers after loading to memory.

2.2. Multi-frame bundled inference

Inference of acoustic frames takes 60–70% of total computations in our LSTM-RNN-based SPSS implementation. Therefore, it is desirable to reduce the amount of computations at the inference stage. In typical ANN-based SPSS, input linguistic features other than state- and frame-position features are constant within a phoneme [2]. Furthermore, speech is a rather stationary process at 5-ms frame shift and target acoustic frames change slowly across frames. Based on these characteristics of inputs and targets this paper explores the multi-frame inference approach [31]. Figure 2 illustrates the concept of multi-frame inference. Instead of predicting one acoustic frame, multiple acoustic frames are jointly predicted at the same time instance. This architecture allows significant reduction in computation while maintaining the streaming capability.

However, preliminary experiments showed degradation due to mismatch between training and synthesis; consistency between input/target alignments (*e.g.*, $\boldsymbol{x}_2 : \{\boldsymbol{y}_1, \boldsymbol{y}_2\}$ or $\boldsymbol{x}_3 : \{\boldsymbol{y}_2, \boldsymbol{y}_3\}$) cannot be guaranteed at the synthesis stage. This issue can be addressed by data augmentation. Figure 3 shows the data augmentation with different frame offset. From aligned input/target pairs, multiple data sequences can be generated with different starting frame offset. By using these data sequences for training, acoustic LSTM-RNNs will generalize to different possible alignments between inputs and targets.

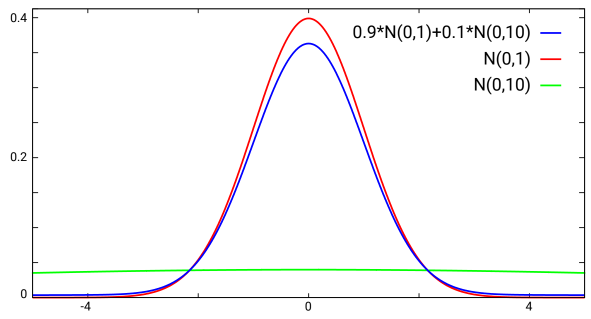

Figure 4: *Plot of 1-dimensional ϵ-contaminated Gaussian distribution ($\boldsymbol{\mu} = [0]$, $\boldsymbol{\Sigma} = [1]$, $\epsilon = 0.1$, $c = 10$).*

2.3. Robust regression

It is known that learning a linear regression model with the squared loss function can suffer from the effect of outliers. Although ANNs trained with the squared loss function are not a simple linear regression model, their output layers perform linear regression given activations at the last hidden layer. Therefore, ANNs trained with the squared loss function can be affected by outliers. These outliers can come from recordings, transcriptions, forced alignments, and F_0 extraction errors.

Using robust regression techniques such as linear regression with a heavy-tailed distribution [32] or minimum density power divergence estimator [33] can relax the effect of outliers. In this work a simple robust regression technique assuming that the errors follow a mixture of two Gaussian distributions, in particular, ϵ-contaminated Gaussian distribution [34] is employed; the majority of observations are from a specified Gaussian distribution, though a small proportion are from a Gaussian distribution with much higher variance, while the two Gaussian distributions share the same mean. The loss function can be defined as

$$\mathcal{L}(\boldsymbol{z}; \boldsymbol{x}, \boldsymbol{\Lambda}) = -\log\big\{(1-\epsilon)\mathcal{N}\left(\boldsymbol{z}; f(\boldsymbol{x}; \boldsymbol{\Lambda}), \boldsymbol{\Sigma}\right) + \epsilon\mathcal{N}\left(\boldsymbol{z}; f(\boldsymbol{x}; \boldsymbol{\Lambda}), c\boldsymbol{\Sigma}\right)\big\}, \quad (1)$$

where $\boldsymbol{z}$ and $\boldsymbol{x}$ denote target and input vectors, $\boldsymbol{\Sigma}$ is a covariance matrix, ϵ and c are weight and scale of outliers, Λ is a set of neural network weights, and $f(\cdot)$ is a non-linear function to predict an output vector given the input vector. Typically, $\epsilon < 0.5$ and $c > 1$. Note that if $\epsilon = 0$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$, the ϵ-contaminated Gaussian loss function is equivalent to the squared loss function. Figure 4 illustrates ϵ-contaminated Gaussian distribution ($\boldsymbol{\mu} = [0]$, $\boldsymbol{\Sigma} = [1]$, $c = 10$ and $\epsilon = 0.1$). It can be seen from the figure that the ϵ-contaminated Gaussian distribution has heavier tail than the Gaussian distribution. As outliers will be captured by the Gaussian distribution with wider variances, the estimation of means is less affected by these outliers. Here using the ϵ-contaminated Gaussian loss function as a criterion to train LSTM-RNNs is investigated for both acoustic and duration LSTM-RNNs. Note that the ϵ-contaminated Gaussian distribution is similar to globally tied distribution (GTD) in [35].

3. Experiments

3.1. Experimental conditions

Speech data from a female professional speaker was used to train speaker-dependent acoustic and duration unidirectional LSTM-RNNs for each language. The configuration for speech analysis stage and data preparation process were the same as those described in [9] except the use of speech at 22.05 kHz sampling rather than 16 kHz and 7-band aperiodicities rather than 5-band ones.

Both the input and target features were normalized to be zero-mean unit-variance in advance. The architecture of the acoustic LSTM-RNNs was 1×128-unit ReLU [36] layer followed by 3×128-cell LSTMP layers [37] with 64 recurrent projection units with a linear recurrent output layer [9]. The duration LSTM-RNN used a single LSTM layer with 64 cells with feed-forward output layer with linear activation. The same model architecture and hyper-parameters were used across all languages.

At the synthesis stage, durations and acoustic features were predicted from linguistic features using the trained networks. Spectral enhancement based on post-filtering in the cepstral domain [25] was applied to improve the naturalness of the synthesized speech. From the acoustic features, speech waveforms were synthesized using the Vocaine vocoder [38].

To subjectively evaluate the performance of the systems, preference tests were also conducted. 100 utterances not included in the training data were used for evaluation. Each pair was evaluated by eight native speaker of each language. The subjects who did not use headphones were excluded from the experimental results. After listening to each pair of samples, the subjects were asked to choose their preferred one, or they could choose "no preference" if they did not have any preference. Note that stimuli that achieved a statistically significant preference ($p < 0.01$) are presented in bold characters in tables displaying experimental results in this section.

3.2. Experimental results for optimizations

3.2.1. Weight quantization

Table 1 shows the preference test result comparing LSTM-RNNs with and without weight quantization. It can be seen from the table that the effect of quantization was negligible. The disk footprint of the acoustic LSTM-RNN for English (NA) was reduced from 1.05 MBytes to 272 KBytes.

3.2.2. Multi-frame inference

While training multi-frame LSTM-RNNs, the learning rate needed to be reduced as mentioned in [31]. Table 2 shows the preference test result comparing single and multi-frame inference. Note that weights of the LSTM-RNNs were quantized to 8-bit integers. It can be seen from the table that LSTM-RNN with multi-frame inference with data augmentation achieved the same naturalness as that with single-frame one. Compared with `1-frame`, `4-frame` achieved about 40% reduction of wall-time at runtime synthesis.

3.2.3. ϵ-contaminated Gaussian loss function

Although c, ϵ, and $\mathbf{\Sigma}$ could be trained with the network weights, they were fixed to $c = 10$, $\epsilon = 0.1$, and $\mathbf{\Sigma} = \boldsymbol{I}$ for both acoustic and duration LSTM-RNNs.Therefore, the numbers of parameters of the LSTM-RNNs trained with the squared and ϵ-contaminated Gaussian loss functions were identical. For training LSTM-RNNs with the ϵ-contaminated Gaussian loss function, the learning rate could be increased. From a few preliminary experiments, the ϵ-contaminated Gaussian loss function with a 2-block structure was selected; 1) mel-cepstrum and aperiodicities, 2) $\log F_0$ and voiced/unvoiced binary flag. This is similar to the multi-stream HMM structure [39] used in HMM-based speech synthesis [25]. Table 3 shows the preference test result comparing the squared and ϵ-contaminated normal loss function to train LSTM-RNNs. Note that all weights of the LSTM-RNNs were quantized to 8-bit integers and 4-frame bundled inference was used. It can be seen from the table that LSTM-RNN trained with the ϵ-contaminated normal loss function achieved the same or better naturalness than those with the squared loss function.

Table 1: *Subjective preference scores (%) between LSTM-RNNs with (`int8`) and without (`float`) 8-bit quantization. Note that "English (GB)", "English (NA)", and "Spanish (ES)" indicate British English, North American English, and European Spanish, respectively.*

Language	`int8`	`float`	No pref.
English (GB)	13.0	12.2	74.8
English (NA)	8.0	10.0	82.0
French	4.7	3.8	91.5
German	12.5	8.8	78.7
Italian	12.0	9.8	78.2
Spanish (ES)	8.8	7.5	83.7

Table 2: *Subjective preference scores (%) between LSTM-RNNs using 4-frame bundled inference with data augmentation (`4-frame`) and single-frame inference (`1-frame`).*

Language	`4-frame`	`1-frame`	No pref.
English (GB)	25.7	20.2	54.2
English (NA)	8.5	6.2	85.3
French	18.8	18.6	62.6
German	19.3	22.2	58.5
Italian	13.5	14.4	72.1
Spanish (ES)	12.8	17.0	70.3

Table 3: *Subjective preference scores (%) between LSTM-RNNs trained with the ϵ-contaminated Gaussian (`CG`) and squared (`L2`) loss functions.*

Language	`CG`	`L2`	No pref.
English (GB)	**27.4**	18.1	54.5
English (NA)	7.6	6.8	85.6
French	**24.6**	15.9	59.5
German	17.1	20.8	62.1
Italian	**16.0**	10.6	73.4
Spanish (ES)	16.0	13.4	70.6

3.3. Comparison with HMM-based SPSS

The next experiment compared HMM- and LSTM-RNN-based SPSS with the optimizations described in this paper. Both HMM- and LSTM-RNN-based acoustic and duration models were quantized into 8-bit integers. The same training data and text processing front-end modules were used.

The average disk footprints of HMMs and LSTM-RNNs

Table 4: *Average latency and total time in milliseconds to synthesize a character, word, sentence, and paragraph by the LSTM-RNN- (`LSTM`) and HMM-based (`HMM`) SPSS systems.*

	Latency (ms)		Total (ms)	
Length	`LSTM`	`HMM`	`LSTM`	`HMM`
char.	12.5	19.5	49.8	49.6
word	14.6	25.3	61.2	80.5
sent.	31.4	55.4	257.3	286.2
para.	64.1	117.7	2216.1	2400.8

Table 5: *Subjective preference scores (%) between the LSTM-RNN- and HMM-based SPSS systems .*

Language	`LSTM`	`HMM`	No pref.
English (GB)	31.6	28.1	40.3
English (NA)	**30.6**	15.9	53.5
French	**68.6**	8.4	23.0
German	**52.8**	19.3	27.9
Italian	**84.8**	2.9	12.3
Spanish (ES)	**72.6**	10.6	16.8

including both acoustic and duration models over 6 languages were 1560 and 454.5 KBytes, respectively. Table 4 shows the average latency (time to get the first chunk of audio) and average total synthesis time (time to get the entire audio) of the HMM and LSTM-RNN-based SPSS systems (North American English) to synthesize a character, word, sentence, and paragraph on a Nexus 6 phone. Note that the execution binary was compiled for modern ARM CPUs having the NEON advanced single instruction, multiple data (SIMD) instruction set [40]. To reduce the latency of the HMM-based SPSS system, the recursive version of the speech parameter generation algorithm [28] with 10-frame lookahead was used. It can be seen from the table that the LSTM-RNN-based system could synthesize speech with lower latency and total synthesis time than the HMM-based system. However, it is worthy noting that the LSTM-RNN-based system was 15–22% slower than the HMM-based system in terms of the total synthesis time on old devices having ARM CPUs without the NEON instruction set (latency was still lower). Table 5 shows the preference test result comparing the LSTM-RNN- and HMM-based SPSS systems. It can be seen from the table that the LSTM-RNN-based system could synthesize more naturally sounding synthesized speech than the HMM-based system.

3.4. Comparison with concatenative TTS

The last experiment evaluated the HMM-driven unit selection TTS [41] and LSTM-RNN-based SPSS with the optimizations described in this paper except quantization. Both TTS systems used the same training data and text processing front-end modules. Note that additional linguistic features which were only available with the server-side text processing front-end modules were used in both systems. The HMM-driven unit selection TTS systems were built from speech at 16 kHz sampling. Although LSTM-RNNs were trained from speech at 22.05 kHz sampling, speech at 16 kHz sampling was synthesized at runtime using a resampling functionality in Vocaine [38]. These LSTM-RNNs had the same network architecture as the one described in the previous section. They were trained with the ϵ-contaminated Gaussian loss function and utilized 4-frame bundled inference. Table 6 shows the preference test result. It can be seen from the table that the LSTM-RNN-based SPSS systems were preferred to the HMM-driven unit selection TTS systems in 10 of 26 languages, while there was no significant preference between them in 3 languages. Note that the LSTM-RNN-based SPSS systems were 3–10% slower but 1,500–3,500 times smaller in disk footprint than the HMM-driven unit selection TTS systems.

Table 6: *Subjective preference scores (%) between the LSTM-RNN-based SPSS and HMM-driven unit selection TTS (`Hybrid`) systems. Note that "Spanish (NA)" and "Portuguese (BR)" indicate North American Spanish and Brazilian Portuguese, respectively.*

Language	`LSTM`	`Hybrid`	No pref.
Arabic	13.9	**22.1**	64.0
Cantonese	**25.1**	7.3	67.6
Danish	37.0	**49.1**	13.9
Dutch	29.1	**46.8**	24.1
English (GB)	22.5	**65.1**	12.4
English (NA)	23.3	**61.8**	15.0
French	28.4	**50.3**	21.4
German	20.8	**58.5**	20.8
Greek	**42.5**	21.4	36.1
Hindi	42.5	36.4	21.1
Hungarian	**56.5**	30.3	13.3
Indonesian	18.9	**57.8**	23.4
Italian	28.1	**49.0**	22.9
Japanese	**47.4**	28.8	23.9
Korean	**40.6**	25.8	33.5
Mandarin	**48.6**	17.5	33.9
Norwegian	**54.1**	30.8	15.1
Polish	14.6	**75.3**	10.1
Portuguese (BR)	31.4	37.8	30.9
Russian	26.7	**49.1**	24.3
Spanish (ES)	21.0	**47.1**	31.9
Spanish (NA)	22.5	**55.6**	21.9
Swedish	**48.3**	33.6	18.1
Thai	**71.3**	8.8	20.0
Turkish	**61.3**	20.8	18.0
Vietnamese	30.8	30.8	38.5

4. Conclusions

This paper investigated three optimizations of LSTM-RNN-based SPSS for deployment on mobile devices; 1) Quantizing LSTM-RNN weights to 8-bit integers reduced disk footprint by 70%, with no significant difference in naturalness; 2) Using multi-frame inference reduced CPU use by 40%, again with no significant difference in naturalness; 3) For training, using an ϵ-contaminated Gaussian loss function rather than a squared loss function to avoid excessive effects from outliers proved beneficial, allowing for an increased learning rate and improving naturalness. The LSTM-RNN-based SPSS systems with these optimizations surpassed the HMM-based SPSS systems in speed, latency, disk footprint, and naturalness on modern mobile devices. Experimental results also showed that the LSTM-RNN-based SPSS system with the optimizations could match the HMM-driven unit selection TTS systems in naturalness in 13 of 26 languages.

5. References

[1] H. Zen, K. Tokuda, and A. Black, "Statistical parametric speech synthesis," *Speech Commn.*, vol. 51, no. 11, pp. 1039–1064, 2009.

[2] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. ICASSP*, 2013, pp. 7962–7966.

[3] H. Lu, S. King, and O. Watts, "Combining a vector space representation of linguistic context with a deep neural network for text-to-speech synthesis," in *Proc. ISCA SSW8*, 2013, pp. 281–285.

[4] Y. Qian, Y. Fan, W. Hu, and F. Soong, "On the training aspects of deep neural network (DNN) for parametric TTS synthesis," in *Proc. ICASSP*, 2014, pp. 3857–3861.

[5] T. Raitio, H. Lu, J. Kane, A. Suni, M. Vainio, S. King, and P. Alku, "Voice source modelling using deep neural networks for statistical parametric speech synthesis," in *Proc. EUSIPCO*, 2014, pp. 2290–2294.

[6] X. Yin, M. Lei, Y. Qian, F. Soong, L. He, Z.-H. Ling, and L.-R. Dai, "Modeling DCT parameterized F0 trajectory at intonation phrase level with DNN or decision tree," in *Proc. Interspeech*, 2014, pp. 2273–2277.

[7] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *Proc. ICASSP*, 2014, pp. 3872–3876.

[8] Y. Fan, Y. Qian, and F. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *Proc. Interspeech*, 2014, pp. 1964–1968.

[9] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *Proc. ICASSP*, 2015, pp. 4470–4474.

[10] R. Fernandez, A. Rendel, B. Ramabhadran, and R. Hoory, "Prosody contour prediction with long short-term memory, bidirectional, deep recurrent neural networks," in *Proc. Interspeech*, 2014, pp. 2268–272.

[11] Z. Wu and S. King, "Investigating gated recurrent neural networks for speech synthesis," in *Proc. ICASSP*, 2016.

[12] Q. Hu, Z. Wu, K. Richmond, J. Yamagishi, Y. Stylianou, and R. Maia, "Fusion of multiple parameterisations for DNN-based sinusoidal speech synthesis with multi-task learning," in *Proc. Interspeech*, 2015, pp. 854–858.

[13] Z. Wu and S. King, "Minimum trajectory error training for deep neural networks, combined with stacked bottleneck features," in *Proc. Interspeech*, 2015.

[14] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King, "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis," in *Proc. ICASSP*, 2015, pp. 4460–4464.

[15] O. Watts, Z. Wu, and S. King, "Sentence-level control vectors for deep neural network speech synthesis," in *Proc. Interspeech*, 2015, pp. 2217–2221.

[16] Y. Fan, Y. Qian, F. Soong, and L. He, "Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis," in *Proc. ICASSP*, 2015, pp. 4475–4479.

[17] F.-L. Xie, Y. Qian, Y. Fan, F. Soong, and H. Li, "Sequence error (SE) minimization training of neural network for voice conversion," in *Proc. Interspeech*, 2014, pp. 2283–2287.

[18] Z. Chen and K. Yu, "An investigation of implementation and performance analysis of DNN based speech synthesis system," in *Proc. ICSP*, 2014, pp. 577–582.

[19] K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "The effect of neural networks in statistical parametric speech synthesis," in *Proc. ICASSP*, 2015, pp. 4455–4459.

[20] B. Uria, I. Murray, S. Renals, and C. Valentini-Botinhao, "Modelling acoustic feature dependencies with artificial neural networks: Trajectory-RNADE," in *Proc. ICASSP*, 2015, pp. 4465–4469.

[21] H. Zen, "Deep learning in speech synthesis," `http://research.google.com/pubs/archive/41539.pdf`, Invited keynote given at ISCA SSW8 2013.

[22] O. Watts, G. Henter, T. Merritt, Z. Wu, and S. King, "From HMMs to DNNs: where do the improvements come from?," in *Proc. ICASSP*, 2016.

[23] A. Robinson and F. Fallside, "Static and dynamic error propagation networks with application to speech coding," in *Proc. NIPS*, 1988, pp. 632–641.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] T. Yoshimura, *Simultaneous modeling of phonetic and prosodic parameters, and characteristic conversion for HMM-based text-to-speech systems*, Ph.D. thesis, Nagoya Institute of Technology, 2002.

[26] H. Zen, "Acoustic modeling for speech synthesis: from HMM to RNN," `http://research.google.com/pubs/pub44630.html`, Invited talk given at ASRU 2015.

[27] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *Proc. ICASSP*, 2000, pp. 1315–1318.

[28] K. Koishida, K. Tokuda, T. Masuko, and T. Kobayashi, "Vector quantization of speech spectral parameters using statistics of dynamic features," in *Proc. ICSP*, 1997, pp. 247–252.

[29] A. Gutkin, J. Gonzalvo, S. Breuer, and P. Taylor, "Quantized HMMs for low footprint text-to-speech synthesis," in *Proc. Interspeech*, 2010, pp. 837–840.

[30] R. Alvarez, R. Prabhavalkar, and A. Bakhtin, "On the efficient representation and execution of deep acoustic models," in *Proc. Interspeech (accepted)*, 2016.

[31] V. Vanhoucke, M. Devin, and G. Heigold, "Multiframe deep neural networks for acoustic modeling," in *Proc. ICASSP*. IEEE, 2013, pp. 7582–7585.

[32] D. Hsu and S. Sabato, "Loss minimization and parameter estimation with heavy tails," *arXiv:1307.1827*, 2013.

[33] G. Henter, S. Ronanki, O. Watts, M. Wester, Z. Wu, and S. King, "Robust TTS duration modeling using DNNs," in *Proc. ICASSP*, 2016.

[34] J. Tukey, "A survey of sampling from contaminated distributions," *Contributions to probability and statistics*, vol. 2, pp. 448–485, 1960.

[35] K. Yu, T. Toda, M. Găsı́c, S. Keizer, F. Mairesse, B. Thomson, and S. Young, "Probablistic modelling of F0 in unvoiced regions in HMM based speech synthesis," in *Proc. ICASSP*, 2009, pp. 3773–3776.

[36] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q.-V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. Hinton, "On rectified linear units for speech processing," in *Proc. ICASSP*, 2013, pp. 3517–3521.

[37] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, 2014, pp. 338–342.

[38] Y. Agiomyrgiannakis, "Vocaine the vocoder and applications is speech synthesis," in *Proc. ICASSP*, 2015, pp. 4230–4234.

[39] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X.-Y. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, "The hidden Markov model toolkit (HTK) version 3.4," `http://htk.eng.cam.ac.uk/`, 2006.

[40] V. Reddy, "NEON technology introduction," *ARM Corporation*, 2008.

[41] J. Gonzalvo, S. Tazari, C.-A. Chan, M. Becker, A. Gutkin, and H. Silen, "Recent advances in Google real-time HMM-driven unit selection synthesizer," in *Proc. Interspeech (accepted)*, 2016.