Merlin: An Open Source Neural Network Speech Synthesis System

Merlin: An Open Source Neural Network Speech Synthesis System

Zhizheng Wu Oliver Watts Simon King

The Centre for Speech Technology Research, University of Edinburgh, United Kingdom

Abstract

We introduce the Merlin speech synthesis toolkit for neural network-based speech synthesis. The system takes linguistic features as input, and employs neural networks to predict acoustic features, which are then passed to a vocoder to produce the speech waveform. Various neural network architectures are implemented, including a standard feedforward neural network, mixture density neural network, recurrent neural network (RNN), long short-term memory (LSTM) recurrent neural network, amongst others. The toolkit is Open Source, written in Python, and is extensible. This paper briefly describes the system, and provides some benchmarking results on a freely-available corpus.

Index Terms: Speech synthesis, deep learning, neural network, Open Source, toolkit

1. Introduction

Text-to-speech (TTS) synthesis involves generating a speech waveform, given textual input. Freely-available toolkits are available for two of the most widely used methods: waveform concatenation [1, for example], and HMM-based statistical parametric speech synthesis, or simply SPSS [2]. Even though the naturalness of good waveform concatenation speech continues to be generally significantly better than that of waveforms generated via SPSS using a vocoder, the advantages of flexibility, control, and small footprint mean that SPSS remains an attractive proposition.

In SPSS, one of the most important factors that limits the naturalness of the synthesised speech [2, 3] is the so-called acoustic model, which learns the relationship between linguistic and acoustic features: this is a complex and non-linear regression problem. For the past decade, hidden Markov models (HMMs) have dominated acoustic modelling [4]. The way that the HMMs are parametrised is critical, and almost universally this entails clustering (or 'tying') groups of models for acoustically- and linguistically-related contexts, using a regression tree. However, the necessary across-context averaging considerably degrades the quality of synthesised speech [3]. One might reasonably say that HMM-based SPSS would be more accurately called regression tree-based SPSS, and then the obvious question to ask is: why not use a more powerful regression model than a tree?

Recently, neural networks have been 'rediscovered' as acoustic models for SPSS [5, 6]. In the 1990s, neural networks had already been used to learn the relationship between linguistic and acoustic features [7, 8, 9], as duration models to predict segment durations [10], and to extract linguistic features from raw text input [11]. The main differences between today and the 1990s are: more hidden layers, more training data, more advanced computational resource, more advanced training algorithms, and significant advancements in the various other techniques needed for a complete parametric speech synthesiser: the vocoder, and parameter compensation/enhancement/postfiltering techniques.

1.1. Recent work neural network speech synthesis

In the recent studies, restricted Boltzmann machines (RBMs) were used to replace Gaussian mixture models to model the distribution of acoustic features [12]. The work claims that RBMs can model spectral details, and result in better quality of synthesised speech. In [13, 14], deep belief networks (DBNs) as deep generative model were employed to model the relationship between linguistic and acoustic features jointly. Deep mixture density networks [15] and trajectory real-valued neural autoregressive density estimators [16] were also employed to predict the probability density function over acoustic features.

Deep feedforward neural networks (DNNs) as a deep conditional model are the model popular model in the literature to map linguistic features to acoustic features directly [17, 18, 19, 20, 21]. The DNNs can be viewed as replacement for the decision tree used in the HMM-based speech as detailed in [22]. It can also be used to model high-dimensional spectra directly [23]. In the feedforward framework, several techniques, such multitask learning [20], minimum generation error [24, 25, 26], have been applied to improve the performance. However, DNNs perform the mapping frame by frame without considering contextual constraints, even though stacked bottleneck features can include some short-term contextual information [26].

To include contextual constraints, a bidirectional long short-term memory (LSTM) based recurrent neural network (RNN) was employed in [27] to formulate TTS as a sequence to sequence mapping problem, that is to map a sequence of linguistic features to the corresponding sequence of acoustic features. In [28], LSTM with a recurrent output layer was proposed to include contextual constraints. In [29], LSTM and gated recurrent unit (GRU) based RNNs are combined with mixture density model to predict a sequence of probability density functions. In [30], a systematic analysis of LSTM-based RNN was presented to provide a better understanding of LSTM.

1.2. The need for a new toolkit

Recently, even though there has been an explosion in the use of neural networks for speech synthesis, a truly Open Source toolkit is missing. Such a toolkit would underpin reproducible research and allow for more accurate cross-comparisons of competing techniques, in very much the same way that the HTS toolkit has done for HMM-based work. In this paper, we intro-

duce Merlin[1], which is an Open Source neural network based speech synthesis system. The system has already been extensively used for the work reported in a number of recent research papers[30, 26, 22, 20, 31, 32, 23, 33, for example]. This paper will briefly introduce the design and implementation of the toolkit and provide benchmarking results on a freely-available speech corpus.

In addition to the results here and in the above list of previously-published papers, Merlin is the DNN benchmark system for the 2016 Blizzard Challenge. There, it is used in combination with the Ossian front-end [2] and the WORLD vocoder [34], both of which are also Open Source and can be used without restriction, to provide an easily-reproducible system.

2. Design and Implementation

Like HTS, Merlin is not a complete TTS system. It provides the core acoustic modelling functions: linguistic feature vectorisation, acoustic and linguistic feature normalisation, neural network acoustic model training, and generation. Currently, the waveform generation module supports two vocoders: STRAIGHT [35] and WORLD [34] but the toolkit is easily extensible to other vocoders in the future. It is equally easy to interface to different front-end text processors.

Merlin is written in Python, based on the theano library. It comes with documentation for the source code and a set of 'recipes' for various system configurations.

2.1. Front-End

Merlin requires an external front-end, such as Festival or Ossian. The front-end output must currently be formatted as HTS-style labels with state-level alignment. The toolkit converts such labels into vectors of binary and continuous features for neural network input. The features are derived from the label files using HTS-style questions. It is also possible to directly provide already-vectorised input features if this HTS-like workflow is not convenient.

2.2. Vocoder

Currently, the system supports two vocoders: STRAIGHT (the C language version) and WORLD. STRAIGHT cannot be included in the distribution because it is not Open Source, but the Merlin distribution does include a modified version of the WORLD vocoder. The modifications add separate analysis and synthesis executables, as is necessary for SPSS. It is not difficult to support some other vocoder, and details on how to do this can be found in the included documentation.

2.3. Feature normalisation

Before training a neural network, it is important to normalise features. The toolkit supports two normalisation methods: min-max, and mean-variance. The min-max normalisation will normalise features to the range of [0.01 0.99], while the mean-variance normalisation will normalise features to zero mean and unit variance. Currently, by default the linguistic features undergo min-max normalisation, while output acoustic features have mean-variance normalisation applied.

[1]The toolkit can be checked out anonymously from the Github repository: https://github.com/CSTR-Edinburgh/merlin

[2]http://simple4all.org/product/ossian

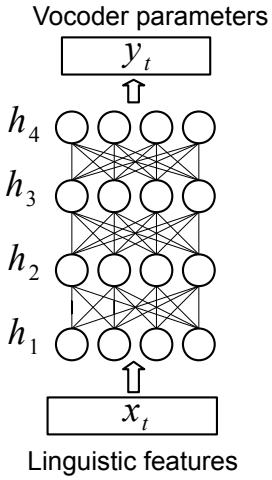

Figure 1: An illustration of feedforward neural network with four hidden layers.

2.4. Acoustic modelling

Merlin includes implementations of several currently-popular acoustic models, each of which comes with an example 'recipe' to demonstrate its use.

2.4.1. *Feedforward neural network*

A feedforward neural network is the simplest type of network. With enough layers, this architecture is usually called a Deep Neural Network (DNN). The input is used to predict the output via several layers of hidden units, each of which performs a nonlinear function, as follows:

$$\mathbf{h}_t = \mathcal{H}(\mathbf{W}^{\mathbf{xh}}\mathbf{x}_t + \mathbf{b}^h) \quad (1)$$

$$\mathbf{y}_t = \mathbf{W}^{\mathbf{hy}}\mathbf{h}_t + \mathbf{b}^y, \quad (2)$$

where $\mathcal{H}(\cdot)$ is a nonlinear activation function in a hidden layer, $\mathbf{W}^{\mathbf{xh}}$ and $\mathbf{W}^{\mathbf{hy}}$ are the weight matrices, $\mathbf{b}^h$ and $\mathbf{b}^y$ are bias vectors, and $\mathbf{W}^{\mathbf{hy}}\mathbf{h}_t$ is a linear regression to predict target features from the activations in the preceding hidden layer. Fig. 1 is an illustration of a feedforward neural network. It takes linguistic features as input and predicts the vocoder parameters through several hidden layers (in the figure, four hidden layers). In the remainder of this paper, we will use **DNN** to indicate a feedforward neural network of this general type. In the toolkit, sigmoid and hyperbolic tangent activation functions are supported for the hidden layers.

2.4.2. *Long short-term memory (LSTM) based RNN*

In a DNN, linguistic features are mapped to vocoder parameters frame by frame without considering the sequential nature of speech. In contrast, recurrent neural networks (RNNs) are

designed for sequence-to-sequence mapping. The use of long short-term memory (LSTM) units is a popular way to realise an RNN.

The basic idea of the LSTM was proposed in [36], and is a commonly used architecture for speech recognition [37]. It is formulated as:

$$\mathbf{i}_t = \delta(\mathbf{W}^{\mathrm{i}}\mathbf{x}_t + \mathbf{R}^{\mathrm{i}}\mathbf{h}_{t-1} + \mathbf{p}^{\mathrm{i}} \odot \mathbf{c}_{t-1} + \mathbf{b}^{\mathrm{i}}), \quad (3)$$

$$\mathbf{f}_t = \delta(\mathbf{W}^{\mathrm{f}}\mathbf{x}_t + \mathbf{R}^{\mathrm{f}}\mathbf{h}_{t-1} + \mathbf{p}^{\mathrm{f}} \odot \mathbf{c}_{t-1} + \mathbf{b}^{\mathrm{f}}), \quad (4)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot g(\mathbf{W}^{\mathrm{c}}\mathbf{x}_t + \mathbf{R}^{\mathrm{c}}\mathbf{h}_{t-1} + \mathbf{b}^{\mathrm{c}}), \quad (5)$$

$$\mathbf{o}_t = \delta(\mathbf{W}^{\mathrm{o}}\mathbf{x}_t + \mathbf{R}^{\mathrm{o}}\mathbf{h}_{t-1} + \mathbf{p}^{\mathrm{o}} \odot \mathbf{c}_t + \mathbf{b}^{\mathrm{o}}), \quad (6)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot g(\mathbf{c}_t). \quad (7)$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are the input, forget, and output gates, respectively; $\mathbf{c}_t$ is the so-called memory cell; $\mathbf{h}_t$ is the hidden activation at time t; $\mathbf{x}_t$ is the input signal; $\mathbf{W}^*$, and $\mathbf{R}^*$ are the weight matrices applied on input and recurrent hidden units, respectively; $\mathbf{p}^*$ and $\mathbf{b}^*$ are the peep-hole connections and biases, respectively; $\delta(\cdot)$ and $g(\cdot)$ are the sigmoid and hyperbolic tangent activation functions, respectively; $\odot$ means element-wise product.

Figure 2 presents an illustration of a standard LSTM unit. It passes the input signal and hidden activation of the previous time instance through an input gate, forget gate, memory cell and output gate to produce the activation. In our implementation, the several variants described in [30] are also available.

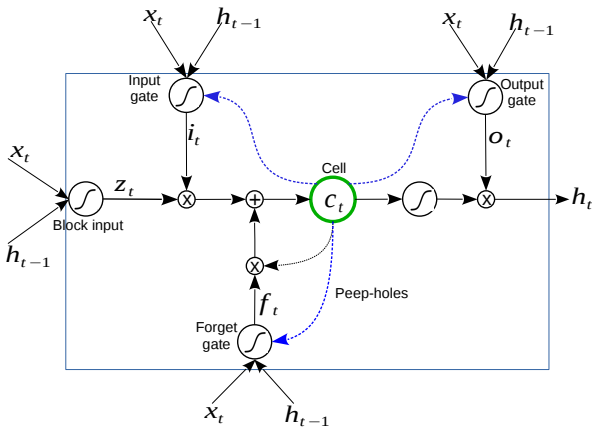

Figure 2: An illustration of a long short-term memory unit. The inputs to the unit are the input signal and the hidden activation of the previous time instance.

2.4.3. Bidirectional RNN

In a uni-directional RNNs, only contextual information from past time instances are taken into account, whilst in a bidirectional RNNs can learn from information propagated both forwards and backwards in time. A bidirectional RNN can be defined as,

$$\overrightarrow{\mathbf{h}}_t = \mathcal{H}(\mathbf{W}^{\mathrm{x}\overrightarrow{\mathrm{h}}}\mathbf{x}_t + \mathbf{R}^{\overrightarrow{\mathrm{h}}\overrightarrow{\mathrm{h}}}\overrightarrow{\mathbf{h}}_{t-1} + \mathbf{b}_{\overrightarrow{\mathrm{h}}}), \quad (8)$$

$$\overleftarrow{\mathbf{h}}_t = \mathcal{H}(\mathbf{W}^{\mathrm{x}\overleftarrow{\mathrm{h}}}\mathbf{x}_t + \mathbf{R}^{\overleftarrow{\mathrm{h}}\overleftarrow{\mathrm{h}}}\overleftarrow{\mathbf{h}}_{t-1} + \mathbf{b}_{\overleftarrow{\mathrm{h}}}), \quad (9)$$

$$\mathbf{y}_t = \mathbf{W}^{\overrightarrow{\mathrm{h}}\mathrm{y}}\overrightarrow{\mathbf{h}}_t + \mathbf{W}^{\overleftarrow{\mathrm{h}}\mathrm{y}}\overleftarrow{\mathbf{h}}_t + \mathbf{b}^{y}, \quad (10)$$

where $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$ are hidden activations from positive and negative directions, respectively; $\mathbf{W}^{\mathrm{x}\overrightarrow{\mathrm{h}}}$ and $\mathbf{W}^{\mathrm{x}\overleftarrow{\mathrm{h}}}$ are weight matrices for input signal; and $\mathbf{R}^{\overrightarrow{\mathrm{h}}\overrightarrow{\mathrm{h}}}$ and $\mathbf{R}^{\overleftarrow{\mathrm{h}}\overleftarrow{\mathrm{h}}}$ are the recurrent matrices for forward and backward directions, respectively.

In bidirectional RNNs, the hidden units can be without gating, or gated units such as LSTM. We will use BLSTM to denote a bidirectional LSTM-based RNN.

2.4.4. Other variants

In Merlin, other variants of neural networks are also implemented, such as gated recurrent units (GRUs) [38], simplified LSTM [30], and the other variants on LSTMs and GRUs described in [30]. All these basic units can be assembled together to create a new architecture by simply changing a configuration file. For example, to implement a 4-layer feedforward neural network using hyperbolic tangent units, one can simply specify the following architecture in the configuration file:

$$[\mathrm{TANH}, \mathrm{TANH}, \mathrm{TANH}, \mathrm{TANH}].$$

Similarly, a hybrid bidirectional LSTM-based RNN can be specified as:

$$[\mathrm{TANH}, \mathrm{TANH}, \mathrm{TANH}, \mathrm{BLSTM}]$$

in the configuration file. More details of the supported unit type can be found in the documentation of the system.

3. Benchmarking performance

3.1. Experimental setup

To demonstrate the performance of the toolkit, we report benchmarking experiments for several architectures implemented in Merlin. A freely-available corpus[3] from a British male professional speaker was used in the experiments. The speech signal was used at a sampling rate of 48 kHz. 2400 utterances were used for training, 70 as a development set, and 72 as the evaluation set. All sets are disjoint.

The front-end for all experiments is Festival. The input features for all neural networks consisted of 491 features. 482 of these were derived from linguistic context, inlcuding quinphone identity, part-of-speech, and positional information within a syllable, word and phrase, etc. The remaining 9 are within-phone positional information: frame position within HMM state and phone, state position within phone both forward and backward, and state and phone durations. The frame alignment and state information was obtained from forced alignment using a monophone HMM-based system with 5 emitting states per phone.

We used two vocoders in these experiments: STRAIGHT [35] and WORLD [34]. STRAIGHT (C language version), which is not Open Source, was used to extract 60-dimensional Mel-Cepstral Coefficients (MCCs), 25 band aperiodicities (BAPs), and fundamental frequency on log scale ($\log F_0$) at 5 msec frame intervals. Similar, WORLD[4], which is Open Source, was also used to extract 60-dimensional MCCs, 5-dimensional BAPs, and $\log F_0$ at 5 msec frame intervals. The

[3] http://dx.doi.org/10.7488/ds/140

[4] The modified version mentioned earlier, and included in the Merlin distribution.

Table 1: Comparison of objective results using the STRAIGHT vocoder. MCD: Mel-Cepstral Distortion. BAP: distortion of band aperiodicities. F0 RMSE is calculated on a linear scale. V/UV: voiced/unvoiced error.

	MCD (dB)	BAP (dB)	F0 RMSE (Hz)	V/UV (%)
DNN	4.09	1.94	8.94	4.15
LSTM	4.03	1.93	8.66	3.98
BLSTM	4.02	1.93	8.68	4.00
BLSTM-S	4.36	1.97	9.37	4.39

Table 2: Comparison of objective results using the WORLD vocoder. MCD: Mel-Cepstral Distortion. BAP: distortion of band aperiodicities. F0 RMSE is calculated on a linear scale. V/UV: voiced/unvoiced error.

	MCD (dB)	BAP (dB)	F0 RMSE (Hz)	V/UV (%)
DNN	4.54	0.36	9.57	11.38
LSTM	4.52	0.35	9.51	11.02
BLSTM	4.51	0.35	9.57	11.18
BLSTM-S	4.70	0.36	10.01	11.66

output features of neural networks thus consisted of MCCs, BAPs, and $\log F_0$ with their deltas and delta-deltas, plus a voiced/unvoiced binary feature.

Before training, the input features were normalised using min-max to the range [0.01, 0.99] and output features were normalised to zero mean and unit variance. At synthesis time, Maximum likelihood parameter generation (MLPG) was applied to generate smooth parameter trajectories from the de-normalised neural network outputs, then spectral enhancement in the cepstral domain was applied to the MCCs to enhance naturalness. Speech Signal Processing Toolkit (SPTK[5]) was used to implement the spectral enhancement.

We report four benchmark systems here:

- DNN: 6 feedforward hidden layers; each hidden layer has 1024 hyperbolic tangent units.
- LSTM: a hybrid architecture with four feedforward hidden layers of 1024 hyperbolic tangent units each, followed by a single LSTM layer with 512 units.
- BLSTM: a hybrid architecture similar to the LSTM, but replacing the LSTM layer with a BLSTM layer of 384 units.
- BLSTM-S: the architecture is the same as BLSTM; the delta and delta-delta features are omitted from the output feature vectors, and no MLPG is applied; theoretically, the BLSTM architecture should be able to learn to derive delta features during training, and should generate trajectories that are already smooth.

3.2. Objective Results

The objective results of the four systems using the STRAIGHT vocoder are presented in Table 1. It is observed that LSTM and BLSTM achieve better objective results than DNN, as expected. The BLSTM-S that does not use dynamic features during training and does not employ MLPG at generation exhibits much higher objective error than all other architectures.

The objective results of the same four architectures, but this time using the WORLD vocoder, are presented in Table 2. The picture is similar to when using the STRAIGHT vocoder. Note that F0 RMSE and V/UV are not directly comparable between Table 1 and 2, as they use different F0 extractors. For both vocoders, we simply use the default settings provided by the respective tools' creators.

In general, the objective results confirm that LSTM and BLSTM can achieve better objective results than DNN (as expected), but that dynamic features and MLPG are still useful for BLSTM, even though it has a theoretical ability to model the necessary trajectory information.

[5] Available at: http://sp-tk.sourceforge.net/

3.3. Subjective Results

We conducted MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor) listening tests to subjectively evaluate the naturalness of the synthesised speech. We evaluated all the four benchmark systems in two separate MUSHRA tests: one for STRAIGHT and a separate test for the WORLD vocoder.

In each MUSHRA test, there were 30 native British English listeners, and each listeners rated 20 sets that were randomly selected from the evaluation set. In each set, a natural speech with the same linguistic content was also included as the hidden reference. The listeners were instructed to give each stimulus a score between 0 and 100, and to rate one of them in each set as 100, which means natural.

The MUSHRA scores for systems using STRAIGHT are presented in Fig 3. It is observed that LSTM and BLSTM are significantly better than DNN (p-value below 0.01). BLSTM produces slightly more natural speech than LSTM, but the difference is not significant. It is also found that BLSTM is significantly more natural than BLSTM-S, consistent with the objective errors reported above.

The MUSHRA scores for systems using WORLD are presented in Fig 4. The relative differences across systems are similar to the STRAIGHT case.

In general, subjective results are consistent with objective results, and there are similar trends regardless of vocoder. Both objective and and subjective results confirm that LSTM and BLSTM offer better performance than DNN, and that MLPG is still useful for BLSTM.

4. Conclusions

In this paper, we have introduced the Open Source Merlin speech synthesis toolkit, and provided reproducible benchmark results on a corpus. We hope the availability of this system will promote open research on neural network speech synthesis, make comparisons between different neural network configurations easier, and allow researchers to report reproducible results. The toolkit, as released, includes the recipes necessary to reproduce all results in this paper, and results in some of our recent publications. The intention is that future results published (by ourselves or others) using this toolkit will also be accompanied by recipe.

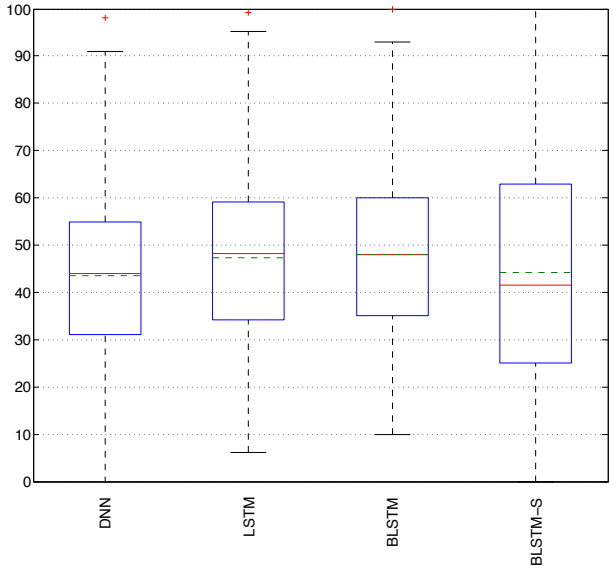

Figure 3: MUSHRA scores for DNN, LSTM, BLSTM, and BLSTM-S using the STRAIGHT vocoder. LSTM and BLSTM are both significantly better than DNN.

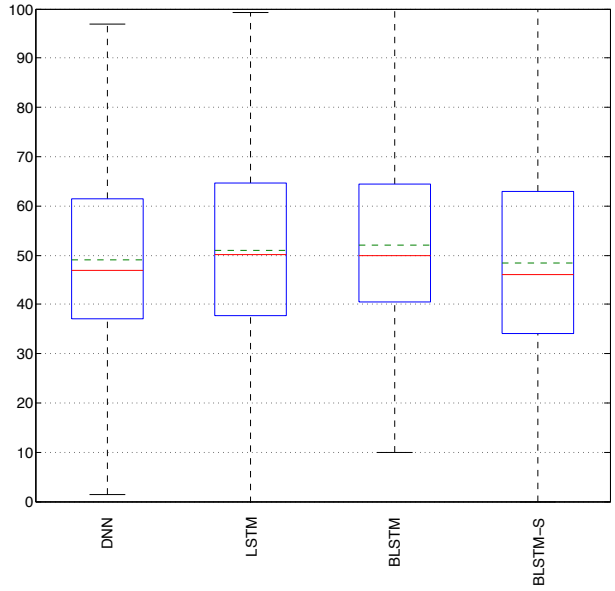

Figure 4: MUSHRA scores for DNN, LSTM, BLSTM, and BLSTM-S using the WORLD vocoder.

Acknowledgement: This work was supported by EPSRC Programme Grant EP/I031022/1 (Natural Speech Technology).

5. References

[1] R. A. J. Clark, K. Richmond, and S. King, "Multisyn: Open-domain unit selection for the Festival speech synthesis system," *Speech Communication*, vol. 49, no. 4, pp. 317–330, 2007.

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

[3] T. Merritt, J. Latorre, and S. King, "Attributing modelling errors in HMM synthesis by stepping gradually from natural to modelled speech," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015, pp. 4220–4224.

[4] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura, "Speech synthesis based on hidden Markov models," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1234–1252, 2013.

[5] Z.-H. Ling, S.-Y. Kang, H. Zen, A. Senior, M. Schuster, X.-J. Qian, H. M. Meng, and L. Deng, "Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 35–52, 2015.

[6] H. Zen, "Acoustic modeling in statistical parametric speech synthesis - from HMM to LSTM-RNN," in *Proc. MLSLP*, 2015, invited paper.

[7] T. Weijters and J. Thole, "Speech synthesis with artificial neural networks," in *Proc. Int. Conf. on Neural Networks*, 1993, pp. 1764–1769.

[8] G. Cawley and P. Noakes, "LSP speech synthesis using backpropagation networks," in *Proc. Third Int. Conf. on Artificial Neural Networks*, 1993, pp. 291–294.

[9] C. Tuerk and T. Robinson, "Speech synthesis using artificial neural networks trained on cepstral coefficients." in *Proc. European Conference on Speech Communication and Technology (Eurospeech)*, 1993, pp. 4–7.

[10] M. Riedi, "A neural-network-based model of segmental duration for speech synthesis," in *Proc. European Conference on Speech Communication and Technology (Eurospeech)*, 1995, pp. 599–602.

[11] O. Karaali, G. Corrigan, N. Massey, C. Miller, O. Schnurr, and A. Mackie, "A high quality text-to-speech system composed of multiple neural networks," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, 1998, pp. 1237–1240.

[12] Z.-H. Ling, L. Deng, and D. Yu, "Modeling spectral envelopes using Restricted Boltzmann Machines and Deep Belief Networks for statistical parametric speech synthesis," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2129–2139, 2013.

[13] S. Kang, X. Qian, and H. Meng, "Multi-distribution deep belief network for speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 8012–8016.

[14] S. Kang and H. Meng, "Statistical parametric speech synthesis using weighted multi-distribution deep belief network," in *Proc. Interspeech*, 2014, pp. 1959–1963.

[15] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014, pp. 3844–3848.

[16] B. Uria, I. Murray, S. Renals, and C. Valentini, "Modelling acoustic feature dependencies with artificial neural networks: Trajectory-rnade," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015, pp. 4465–4469.

[17] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 7962–7966.

[18] 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," *Proc. the 8th ISCA Speech Synthesis Workshop (SSW)*, pp. 281–285, 2013.

[19] Y. Qian, Y. Fan, W. Hu, and F. K. Soong, "On the training aspects of deep neural network (DNN) for parametric TTS synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014, pp. 3829–3833.

[20] 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. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015, pp. 4460–4464.

[21] K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "The effect of neural networks in statistical parametric speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015, pp. 4455–4459.

[22] O. Watts, G. E. Henter, T. Merritt, Z. Wu, and S. King, "From HMMs to DNNs: where do the improvements come from?" in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[23] C. Valentini-Botinhao, Z. Wu, and S. King, "Towards minimum perceptual error training for DNN-based speech synthesis," in *Proc. Interspeech*, 2015, pp. 869–873.

[24] Z. Wu and S. King, "Minimum trajectory error training for deep neural networks, combined with stacked bottleneck features," in *Proc. Interspeech*, 2015, pp. 309–313.

[25] Y. Fan, Y. Qian, F. K. Soong, and L. He, "Sequence generation error (SGE) minimization based deep neural networks training for text-to-speech synthesis," in *Proc. Interspeech*, 2015, pp. 864–868.

[26] Z. Wu and S. King, "Improving trajectory modelling for dnn-based speech synthesis by using stacked bottleneck features and minimum generation error training," *IEEE Trans. Audio, Speech and Language Processing*, 2016.

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

[28] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015, pp. 4470–4474.

[29] B. X. Wenfu Wang, Shuang Xu, "Gating recurrent mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[30] Z. Wu and S. King, "Investigating gated recurrent neural networks for speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[31] T. Merritt, J. Yamagishi, Z. Wu, O. Watts, and S. King, "Deep neural network context embeddings for model selection in rich-context HMM synthesis," in *Proc. Interspeech*, 2015.

[32] T. Merritt, R. A. Clark, Z. Wu, J. Yamagishi, and S. King, "Deep neural network-guided unit selection synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[33] 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.

[34] M. MORISE, F. YOKOMORI, and K. OZAWA, "WORLD: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE transactions on information and systems*, 2016.

[35] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigné, "Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds," *Speech communication*, vol. 27, no. 3, pp. 187–207, 1999.

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

[37] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[38] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.