Minimum trajectory error training for deep neural networks, combined with stacked bottleneck features

Zhizheng Wu Simon King

The Centre for Speech Technology Research, University of Edinburgh, United Kingdom
{zhizheng.wu, simon.king}@ed.ac.uk

Abstract

Recently, Deep Neural Networks (DNNs) have shown promise as an acoustic model for statistical parametric speech synthesis. Their ability to learn complex mappings from linguistic features to acoustic features has advanced the naturalness of synthesis speech significantly. However, because DNN parameter estimation methods typically attempt to minimise the mean squared error of each individual frame in the training data, the dynamic and continuous nature of speech parameters is neglected. In this paper, we propose a training criterion that minimises speech parameter trajectory errors, and so takes dynamic constraints from a wide acoustic context into account during training. We combine this novel training criterion with our previously proposed stacked bottleneck features, which provide wide linguistic context. Both objective and subjective evaluation results confirm the effectiveness of the proposed training criterion for improving model accuracy and naturalness of synthesised speech.
Index Terms: Speech synthesis, acoustic model, deep neural network, trajectory error

1. Introduction

Statistical parametric speech synthesis (SPSS) has been slowly advancing in naturalness, yet can still only generate synthetic speech with 'acceptable' quality. Even though SPSS offers more flexibility and controllability than unit-selection speech synthesis, naturalness is significantly worse than good unit-selection speech, as seen across many years of the Blizzard Challenge.

One limiting factor in the naturalness of SPSS using hidden Markov models (HMMs) is the averaging across different linguistic contexts [1] that is intrinsic to standard decision-tree-clustered HMMs. That is, the accuracy of the acoustic model directly affects the naturalness of the generated speech [2]. The focus of this paper is to improve the accuracy of the acoustic model.

There have been many attempts to improve acoustic models for HMM-based speech synthesis. In [3], a minimum generation error training criteria was proposed to address the inconsistency between training and generation criteria. In [4], a trajectory hidden Markov model was proposed to explicitly model the relationship between static and dynamic features. In [5] and [6], global variance and modulation spectrum enhancement techniques were proposed, respectively. None of these techniques have directly addressed the issue of across-context averaging in decision tree based clustering, which we have found to degrade the quality of synthesised speech considerably [1].

Recently, following the success of Deep Neural Networks (DNNs) as an acoustic model in automatic speech recognition [7], neural networks have re-emerged as an alternative acoustic model for SPSS, and several studies have presented state-of-the-art performance using DNNs [8, 9, 10, 11, 12, 13]. In [8, 13], a feed-forward neural network was used to map linguistic features to acoustic features (i.e., vocoder parameters) directly. In [9], a restricted Boltzmann machine (RBM) was used the replace Gaussian distributions and so to model acoustic feature distributions more precisely. In [10], a deep belief network (DBN) was employed to model the joint probability of linguistic and acoustic features. Generally, DNNs are used to map a set of linguistic features to the corresponding acoustic features, frame by frame.

As speech is a dynamic signal, temporal information is important for both naturalness and intelligibility of the synthetic speech. However, current implementations of DNN-based speech synthesis make a *frame-by-frame independence* assumption during modelling and generation. The frame-by-frame independence assumption has two consequences. The first is the frame-wise independence assumption when predicting acoustic features. Even though contextual information has been included in the linguistic features, when predicting acoustic features for consecutive frames, each frame is generated conditionally independently of the others, given the linguistic context. This has implications for the trajectory of the generated acoustic features. The second consequence is the ignorance of the interaction between static and dynamic features during training DNN models. Dynamic features are extracted from a sequence of static features; but, after this extraction, the relationships between static and dynamic features is neglected during training. This has implications for the accuracy of the acoustic model (i.e., DNN) itself.

In previous work [13], we offered a partial solution to these problems by stacking consecutive frames of bottleneck features, to better capture contextual constraints. These constraints are applied in both training and generation. However, in that work, we still made the usual (in the case of DNN speech synthesis) conditional independence assumption between static and dynamic acoustic features, during both training and generation.

Here, we propose a novel training criterion – minimum trajectory error for DNNs – and we combine this with stacked bottleneck features. The new criterion is inspired by minimum generation error for HMM-based speech synthesis [3] and sequence error minimisation for voice conversion [14]. Rather than minimising frame-wise mean squared error, the minimum trajectory error criterion considers the dynamic feature constraints in the training phase. By integrating this criterion with stacked bottleneck features (which can be viewed as an acoustically-supervised compression and denoising of linguistic context), we can now include contextual constraints at the input linguistic level and the output acoustic level.

2. Problem statement

In this section, we review the two stages in DNN-based speech synthesis – offline training and runtime generation – and then discuss the problems of the current framework. During offline training, a DNN learns the relationship between input linguistic features $\mathbf{x}$ and corresponding output acoustic features $\mathbf{o}$,

$$\mathbf{o} = \mathcal{F}(\mathbf{x}) + \mathbf{e}, \tag{1}$$

where $\mathcal{F}(\cdot)$ is the nonlinear mapping function learned by the DNN, and $\mathbf{e}$ is the modelling error. The acoustic feature $\mathbf{o}$ usually consists of static features $\mathbf{c}$, and corresponding dynamic features $\Delta\mathbf{c}$ and $\Delta^2\mathbf{c}$, as

$$\mathbf{o} = [\mathbf{c}^\top, \Delta\mathbf{c}^\top, \Delta^2\mathbf{c}^\top]^\top \tag{2}$$

Hence, a sequence of the observation acoustic features $\mathbf{O}$ can be written as

$$\mathbf{O} = \mathbf{W}\mathbf{C}, \tag{3}$$

where $\mathbf{W}$ contains the coefficients needed expand the static feature vector sequence $\mathbf{C}$ into sequence $\mathbf{O}$ that also includes delta and delta-delta features [15].

To train a DNN, the objective is to minimise the error between generated $\hat{\mathbf{o}}$ and observed acoustic features $\mathbf{o}$:

$$D(\hat{\mathbf{o}}, \mathbf{o}) = (\hat{\mathbf{o}} - \mathbf{o})^\top (\hat{\mathbf{o}} - \mathbf{o}). \tag{4}$$

Gradient descent algorithms, such as classical back-propagation [16], can be applied to minimise the error function. The gradients of model parameters can be calculated by taking derivatives of $D(\hat{\mathbf{o}}, \mathbf{o})$ with respect to the model parameters λ, as follows:

$$\frac{\partial D(\hat{\mathbf{o}}, \mathbf{o})}{\partial \lambda} = \frac{\partial D(\hat{\mathbf{o}}, \mathbf{o})}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial \lambda}, \tag{5}$$

where

$$\frac{\partial D(\hat{\mathbf{o}}, \mathbf{o})}{\partial \mathbf{o}} = \hat{\mathbf{o}} - \mathbf{o} \tag{6}$$

is the error at the output layer to be back-propagated through the network from the output layer to the input layer, calculating the gradients of model parameters at each layer. In practice, because the error of each feature vector is computed independently, a mini-batch gradient descent method is typically applied for fast computation and stable optimisation performance.

At runtime generation, given a sequence of linguistic features $\mathbf{X}$, the corresponding acoustic features $\hat{\mathbf{O}}$ are generated from the trained DNN frame by frame as $\hat{\mathbf{O}} = \mathcal{F}(\mathbf{X})$. To generate smooth parameter trajectory, maximum likelihood parameter generation (MLPG) algorithm is used by taking dynamic features into account. The MLPG algorithm can be written as:

$$\hat{\mathbf{C}} = (\mathbf{W}^\top \mathbf{U}^{-1} \mathbf{W})^{-1} \mathbf{W}^\top \mathbf{U}^{-1} \hat{\mathbf{O}} \tag{7}$$

where $\hat{\mathbf{C}}$ is the predicted static acoustic feature sequence (i.e., trajectory). As reported in previous work, MLPG with dynamic features is important for good quality speech [17].

Even though dynamic features are used in the MLPG algorithm as a constraint for smooth parameter trajectories, the dynamic features are still treated no differently to the static features during model training and generation: the relationship between static and dynamic features is neglected. Since the objective of the DNN is to generate more accurate parameter trajectories, it should be beneficial to include these dynamic constraints during training. A similar approach is already available for HMM-based synthesis [3, 4].

3. Proposed minimum trajectory error training

To model the interaction between static and dynamic features and include temporal constraints in the training phase, we propose a new training criterion: to minimise the utterance-level trajectory error, rather than the frame-by-frame error. In this way, we minimise the error of the final smoothed trajectory directly, rather than the intermediate features. In other words, we minimise the error of the output of MLPG (which will of course then be used directly to generate speech), rather than minimising the error of the features that are input to MLPG.

The trajectory error function is defined as,

$$D(\hat{\mathbf{C}}, \mathbf{C}) = (\hat{\mathbf{C}} - \mathbf{C})^\top (\hat{\mathbf{C}} - \mathbf{C}) \tag{8}$$

$$= (\mathbf{R}\hat{\mathbf{O}} - \mathbf{C})^\top (\mathbf{R}\hat{\mathbf{O}} - \mathbf{C}), \tag{9}$$

where $\mathbf{C}$ and $\hat{\mathbf{C}}$ are the reference and generated parameter trajectories, respectively, and $\mathbf{R} = (\mathbf{W}^\top \mathbf{U}^{-1} \mathbf{W})^{-1} \mathbf{W}^\top \mathbf{U}^{-1}$ is the matrix to perform parameter generation, given static and delta features. In comparison with Eq. (4), the new error function is computed from the smoothed trajectory rather than the direct output of the DNN. That is, the neural network will model parameter trajectories directly, and hence we need to take the MLPG algorithm into account whilst training the network.

Similar to conventional DNN, gradient descent methods can be used to train the network. The gradients of DNN model parameters λ can computed as:

$$\frac{\partial D(\hat{\mathbf{C}}, \mathbf{C})}{\partial \lambda} = \frac{\partial D(\hat{\mathbf{C}}, \mathbf{C})}{\partial \hat{\mathbf{O}}} \frac{\partial \hat{\mathbf{O}}}{\partial \lambda} \tag{10}$$

$$= \frac{\partial D(\mathbf{R}\mathbf{O}, \mathbf{C})}{\partial \hat{\mathbf{O}}} \frac{\partial \hat{\mathbf{O}}}{\partial \lambda}, \tag{11}$$

where

$$\frac{\partial D(\mathbf{R}\hat{\mathbf{O}}, \mathbf{C})}{\partial \hat{\mathbf{O}}} = (\hat{\mathbf{C}} - \mathbf{C})^\top \mathbf{R} \tag{12}$$

Here only $\frac{\partial \hat{\mathbf{O}}}{\partial \lambda}$ is directly related to the model parameters. In comparison to Eq. (5), the only difference between the new training criterion and the conventional frame-based mean squared error criterion is the method for computing the errors to be back-propagated through the network. The method for computing gradients for DNN parameters in lower layers is not changed.

The training algorithm with the new criterion is similar to conventional mini-batch gradient descent. Because we are considering trajectories, all frames from each training utterance must now be in the same mini-batch (usually, this is not the case: frames are shuffled). Thus, the sizes of mini-batches vary: each mini-batch comprises all frames from a single training utterance. At runtime generation, things proceed exactly as with a conventionally-trained DNN, as per Eq. (7).

In implementation, most of the computational cost arises from the calculation of $(\mathbf{W}^\top \mathbf{U}^{-1} \mathbf{W})^{-1}$. As $\mathbf{U}$ is diagonal, $\mathbf{W}^\top \mathbf{U}^{-1} \mathbf{W}$ becomes a banded matrix, and the computational costs can be reduced considerably. In practice, the errors in the output layer as presented in Eq. (12) are computed dimension by dimension. After the errors of all the dimensions are computed, the back-propagation and gradient update processes are the same as conventional training algorithm. In our implementation, minimum trajectory error training is about 20 times slower than conventional frame-wise error training.

4. Experiments

4.1. Experimental setup

In our experiments, we used a corpus from a British male speaker. The corpus was divided into three subsets, 2400 utterances as training set, 70 utterances as development set, and 72 utterances as testing set. The waveform sampling rate is 48 kHz. The STRAIGHT vocoder [18] was to used extract 60-dimensional Mel-Cepstral Coefficients (MCCs), 25 band aperiodicities (BAPs), and log-scale fundamental frequency ($\log F_0$ at a 5-ms frame step. During synthesis, we used the same vocoder to reconstruct speech.

We have already shown [13] that our DNN-based system is significantly better than our HMM-based system. Hence, we did not include any HMM-based baselines in this experiment. The following four systems were compared (FE=frame error; MTE=minimum trajectory error; BN=bottleneck features):

FE-DNN: This is a feed-forward neural network trained in the conventional way, to minimise frame-by-frame error. The input features consisted of 592 binary features and 9 numerical features (601 in total). The 592 binary features were derived from linguistic context such as quinphone identities, and part-of-speech, position information of phoneme, syllable, word and phrase. The 9 numerical features were frame position information, such as frame position in HMM state and phoneme, state position in phoneme, and state and phoneme durations. The output features consisted of 60-D MCCs, 25-D BAPs, 1-D F_0 and their corresponding delta, delta-delta features. F_0 was interpolated linearly in unvoiced regions, and voiced-unvoiced information was added as an output feature of the network. The network had six hidden layers, each of 1024 units. The bottom layers used tangent activation function, while the output layer was a linear regression layer. A learning rate of 0.02 and a momentum of 0.3 were used in the first 10 epochs, then after 10 epochs the learning rate was halved at each epoch, and momentum was set to 0.9. The maximum number of epochs was 30.

MTE-DNN: This is a feed-forward neural network trained by the minimum trajectory error (MTE) criterion proposed here. The input and output features were the same as for FE-DNN. The network architecture was also the same as FE-DNN. The weights of the MTE-DNN were initialised from the fully converged FE-DNN. Learning rate and momentum were set to 0.02 and 0.6 for the first 10 epochs. After that, momentum was set to 0.9, and the learning rate was halved at each epoch.

FE-BN-DNN: This is similar to the FE-DNN system, but with stacked bottleneck features and linguistic features as input. The only difference between FE-BN-DNN and FE-DNN is the use of stacked bottleneck features. In the bottleneck network, the second hidden layer was set as bottleneck layer, with 32 hidden units. 21 consecutive frames (middle frame $+/-$ 10 frames) of bottleneck features were stacked as input. Hence, the dimension of input layer was 1273 ($32 \times 21 + 601$).

MTE-BN-DNN: This is a DNN with the same input and output features as FE-BN-DNN, but trained with the proposed minimum trajectory error training criterion. It was initialised from the converged FE-BN-DNN. Momentum was set to 0.8 for the first 10 epochs, and after that changed to 0.9. The learning rate was set to 0.04 for the first 10 epochs, and after that it was halved at each epoch.

Note that in all the neural nets, the top two layers used a halved learning rate compared to lower layers. In the implementations of above systems, CUDAMat[1] was employed, to enable training on a GPU.

4.2. Objective evaluation

We first conducted an objective evaluation to assess the performance of the proposed training criterion. Although objective evaluation results do not always correlate well with perceived quality, objective evaluation is still useful for tuning DNN hyper-parameters (such as learning rate, layer sizes, etc), which affect performance considerably.

First, we checked the convergence properties of the proposed training criterion. Fig. 4.2 presents the mean squared error of MTE-BN-DNN on training and development sets as a function of the number of training epochs. Note that error is measured *after* MLPG, since these are the acoustic features that will be used to generate speech. It can be observed that the new criterion reduces the trajectory error, which converges after about 15 iterations on both training and development sets. We note that the error jump at the 11^{th} epoch is normal and expected, since we changed the momentum value at that epoch; this phenomenon was reported in [19].

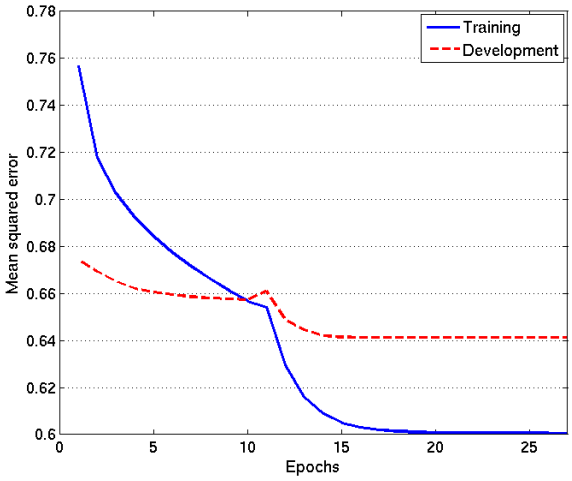

Figure 2: Convergence of minimum trajectory error training for the DNN with stacked bottleneck features (MTE-BN-DNN)

Second, we compared the objective distortions of DNNs with and without the new training criterion on the testing set. The results are presented in Table 1. Compared to FE-DNN, both MCD and F_0 RMSE of MTE-DNN are reduced from 4.19 dB and 9.13 Hz to 4.12 dB and 8.93 Hz, respectively. In comparison with FE-BN-DNN, MTE-DNN considers dynamic feature constraints at output acoustic level by minimising the trajectory errors, while FE-BN-DNN uses temporal constraints at the input linguistic feature level by stacking bottleneck features. The objective results suggest that *only* stacking contextual bottleneck features is more effective than *only* consider-

[1] https://github.com/cudamat/cudamat

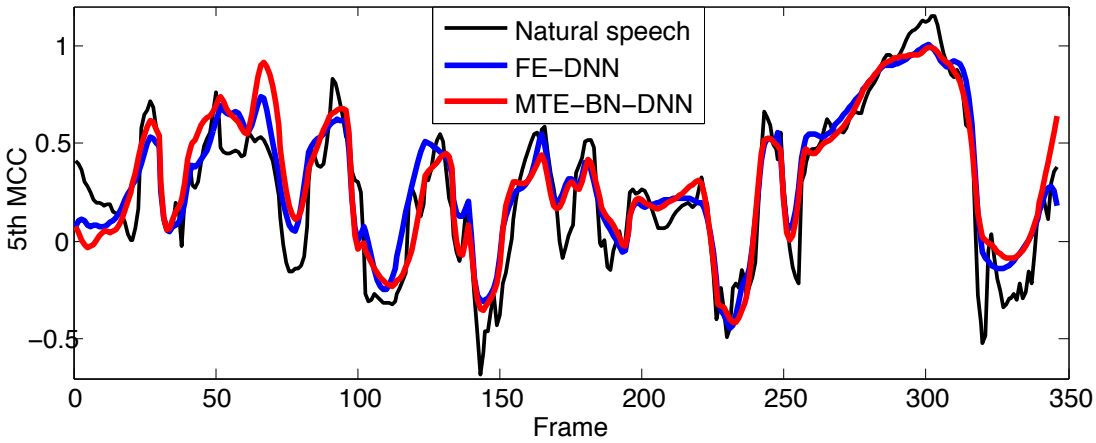

Figure 1: Trajectories of the 5^{th} Mel-Cepstral Coefficient (MCC) of reference natural speech and those predicted by baseline FE-DNN and proposed MTE-BN-DNN systems.

ing dynamic constraints at the output acoustic level. However, combining both gives the best overall performance: MTE-BN-DNN achieves a 0.04 dB reduction in MCD compared to FE-BN-DNN, with similar F_0 RMSE performance.

Table 1: Objective results. MCD = Mel Cepstral Distortion. Root mean squared error (RMSE) of F_0 was computed in linear frequency. V/UV error means frame-level voiced/unvoiced error.

	MCD (dB)	F_0 RMSE (Hz)	V/UV error rate (%)
FE-DNN	4.19	9.13	4.24
MTE-DNN	4.12	8.93	4.28
FE-BN-DNN	4.03	8.91	3.97
MTE-BN-DNN	3.99	8.97	4.02

Fig. 4.2 illustrates the trajectories of the 5^{th} MCC of natural speech and those predicted by the baseline FE-DNN and the proposed MTE-BN-DNN systems. It is observed that both systems can predict reasonable trajectories. The proposed system predicts a trajectory that is on average closer to the reference natural speech. In general, the objective evaluation results confirm the effectiveness of minimum trajectory error as a training criterion, suggesting that acoustic model accuracy can indeed be improved with the proposed method.

4.3. Subjective evaluation

We then conducted a subjective evaluation to assess the naturalness of the synthesised speech via preference tests. We considered four pairs: FE-DNN vs MTE-DNN, FE-BN-DNN vs MTE-BN-DNN, FE-BN-DNN vs MTE-DNN and MTE-BN-DNN vs MTE-DNN. 27 paid native English speakers participated. Each listener was asked to listen 20 randomly selected pairs. In each pair, the listener was asked to listen to pairs of spoken utterances (each half of the pair was the same text generated from differing systems), and then decide which one sounded more natural.

The preference results are presented in Fig. 4.3. First, let us examine the effectiveness of the proposed MTE training criterion. It can be observed that MTE-DNN is significantly better than FE-DNN. MTE-BN-DNN also achieves a slightly a higher preference score than FE-BN-DNN, although the difference is not significant.

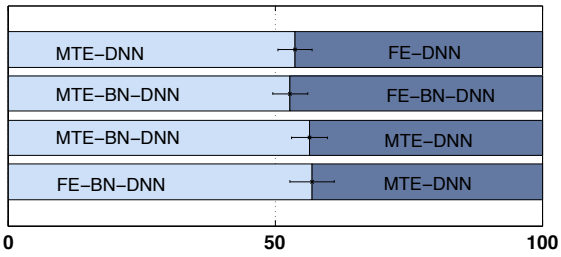

Figure 3: Preference test results for naturalness.

We can also compare the two ways of including temporal constraints by comparing with MTE-DNN and FE-BN-DNN. FE-BN-DNN is significantly better than MTE-DNN in terms of naturalness. This indicates that stacking bottleneck feature at the input level is more effective than considering only temporal constraints at the output acoustic feature level.

Last, we assessed whether the integration of minimum trajectory error criteria and stacked bottleneck features is effective. MTE-BN-DNN is significantly better than MTE-DNN that does not have stacked bottleneck features, and has a slightly (but not significantly) higher listener preference than FE-BN-DNN, which does not use minimum trajectory error criteria. It appears that the minimum trajectory error criterion and stacked bottleneck features approaches are complementary.

5. Conclusions

In this paper, we proposed a new training criterion for DNN-based speech synthesis, which minimises the trajectory error rather than frame-by-frame error. We integrated the training criterion with our previously-proposed stacked bottleneck features and obtained significantly improvement over a baseline DNN system in terms of naturalness.

The samples used in the listening tests are vailable online via: http://datashare.is.ed.ac.uk/handle/10283/786.

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

6. References

[1] 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.

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

[3] Y.-J. Wu and R.-H. Wang, "Minimum generation error training for HMM-based speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2006.

[4] H. Zen, K. Tokuda, and T. Kitamura, "Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences," *Computer Speech & Language*, vol. 21, no. 1, pp. 153–173, 2007.

[5] T. Tomoki and K. Tokuda, "A speech parameter generation algorithm considering global variance for HMM-based speech synthesis," *IEICE Transactions on Information and Systems*, vol. 90, no. 5, pp. 816–824, 2007.

[6] S. Takamichi, T. Toda, G. Neubig, S. Sakti, and S. Nakamura, "A postfilter to modify the modulation spectrum in HMM-based speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[7] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

[15] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 2000.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[17] Y. Chen, Z.-j. Yan, and F. K. Soong, "A perceptual study of acceleration parameters in HMM-based TTS," in *Proc. Interspeech*, 2010.

[18] 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.

[19] G. Hinton, "A practical guide to training restricted Boltzmann machines," University of Toronto, Tech. Rep., 2010.