INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

A Neural Parametric Singing Synthesizer

Merlijn Blaauw, Jordi Bonada

Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
merlijn.blaauw@upf.edu, jordi.bonada@upf.edu

Abstract

We present a new model for singing synthesis based on a modified version of the WaveNet architecture. Instead of modeling raw waveform, we model features produced by a parametric vocoder that separates the influence of pitch and timbre. This allows conveniently modifying pitch to match any target melody, facilitates training on more modest dataset sizes, and significantly reduces training and generation times. Our model makes frame-wise predictions using mixture density outputs rather than categorical outputs in order to reduce the required parameter count. As we found overfitting to be an issue with the relatively small datasets used in our experiments, we propose a method to regularize the model and make the autoregressive generation process more robust to prediction errors. Using a simple multi-stream architecture, harmonic, aperiodic and voiced/unvoiced components can all be predicted in a coherent manner. We compare our method to existing parametric statistical and state-of-the-art concatenative methods using quantitative metrics and a listening test. While naive implementations of the autoregressive generation algorithm tend to be inefficient, using a smart algorithm we can greatly speed up the process and obtain a system that's competitive in both speed and quality.
Index Terms: singing synthesis, machine learning, deep learning, conditional generative models, autoregressive models

1. Introduction

Many of today's more successful singing synthesizers are based on concatenative methods. That is, they transform and concatenate short waveform units selected from an inventory of recordings of a singer. While such systems are the state-of-the-art in terms of sound quality and naturalness [1], they are limited in terms of flexibility and can be difficult to extend or significantly improve upon. On the other hand, machine learning-based approaches, such as statistical parametric methods [2, 3], are much less rigid and do allow for things such as combining data from multiple speakers, model adaptation using small amounts of training data, joint modeling of timbre and expression, etc. Unfortunately, so far these systems have been unable to match the sound quality of concatenative methods, in particular suffering from oversmoothing in frequency and time.

Recent advances in generative models for Text-to-Speech Synthesis (TTS) using Deep Neural Networks (DNNs), in particular the WaveNet model [4], showed that model-based approaches can achieve sound quality on-par or even beyond that of concatenative systems. This model's ability to accurately generate raw speech waveform sample-by-sample, clearly shows that oversmoothing is not an issue. While directly modeling the waveform signal is very attractive, we feel that for singing voice the more traditional approach of using a parametric vocoder is better suited. Compared to speech, the melodic component of singing results in a wider range of pitch and timbre combinations, and thus a larger waveform space. We consider the required amount of training data impractical for our use case. Using a parametric vocoder effectively separates pitch and timbre, thus significantly simplifying the problem of synthesizing any melody. While a vocoder unavoidably introduces some degradation in sound quality, we consider the degradation introduced by current models still a more important factor. Thus, if we can improve the quality of the generative model, we should be able to achieve a quality closer to the upper bound the vocoder can provide, i.e. round-trip vocoder analysis-synthesis.

2. Related work

Our method is heavily based on a class of fully-visible probabilistic autoregressive generative models that use neural networks with similar architectures. This type of model was first proposed to model natural images (PixelCNN) [5, 6, 7], but was later also applied to modeling raw audio waveform (WaveNet) [4], video (Video Pixel Networks) [8] and text (ByteNet) [9].

The SampleRNN [10] model proposes an alternative architecture for unconditional raw waveform generation based on multi-scale hierarchical Recurrent Neural Networks (RNNs) rather than dilated Convolutional Neural Networks (CNNs). Other works [11, 12] have extended these two architectures to include attention mechanisms to allow performing end-to-end TTS, i.e. generation conditioned on unaligned orthographic or phonetic sequences rather than aligned linguistic features.

More traditional neural parametric speech synthesizers tend to be based on feed-forward architectures such as DNNs and Mixture Density Networks (MDNs) [13], or on recurrent architectures such as Long Short-Term Memory RNNs (LSTM-RNNs) [14]. Feed-forward networks learn a frame-wise mapping between linguistic and acoustic features, thus potentially producing discontinuous output. This is often partly mitigated by predicting static, delta and delta-delta feature distributions combined with a parameter generation algorithm that maximizes output probability [15]. Recurrent architectures avoid this issue by propagating hidden states (and sometimes the output state) over time. In contrast, autoregressive architectures like the one we propose make predictions based on predicted past acoustic features, allowing, among other things, to better model rapid modulations such as plosive and trill consonants.

There have been several works proposing different types of singing synthesizers. The more prominent of which are based on concatenative methods [16, 1] and statistical parametric methods centered around Hidden Markov Models (HMMs) [2, 3]. While this work focuses on the generation of timbre without considering pitch or timing, an important difference between these methods is that statistical models allow joint modeling of timbre and musical expression from natural singing [17, 18]. Concatenative methods in contrast typically use disjoint modeling and specialized recordings. Many of the techniques developed for HMM-based TTS are also applicable to singing synthesis, e.g. speaker-adaptive training [19]. The main drawback of HMM-based approaches is that phonemes

http://dx.doi.org/10.21437/Interspeech.2017-1420

are modeled using a small number of discrete states and within each state statistics are constant. This causes excessive averaging, an overly static "buzzy" sound and noticeable state transitions in long sustained vowels in the case of singing. More recently, some work has also been done on using feed-forward DNNs for singing synthesis [20], albeit with a somewhat limited architecture.

3. Proposed system

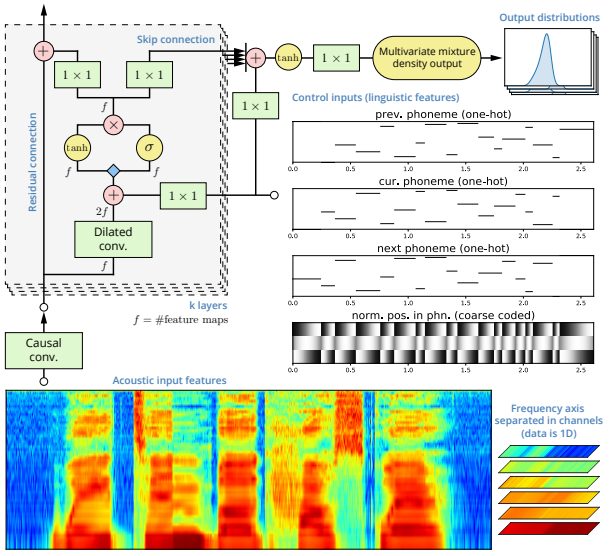

Figure 1: *Overview of the proposed network architecture.*

The architecture of our proposed model, its inputs, and its outputs are summarized in fig. 1. The model consists of a neural network that takes a window of past acoustic features as input and predicts a probability distribution of acoustic features corresponding to the current time step. Additionally, the network has control inputs in the form of linguistic features, which allow controlling when the model generates which phoneme.

Like its predecessors, our model is based on the idea of factorizing a joint probability as a product of conditional probabilities with some causal ordering. The conditional probability distributions are predicted by a neural network trained to maximize likelihood of a observation given past observations. To synthesize, predictions are made by sampling the distribution conditioned on past predictions, that is, in a sequential, autoregressive manner. However, while this factorization is generally done for individual variables (i.e. waveform samples or pixels), we do so for vectors of variables corresponding to a single frame,

$$p\left(\mathbf{x}_1, \ldots, \mathbf{x}_T\right) = \prod_{t=1}^{T} p\left(\mathbf{x}_t \mid \mathbf{x}_{<t}\right), \tag{1}$$

where $\mathbf{x}_t$ is an N-dimensional vector of acoustic features $[x_{t,1}, \ldots, x_{t,N}]$, and T is the length of the signal. In our case we consider the variables within a frame to be conditionally independent,

$$p\left(\mathbf{x}_t \mid \mathbf{x}_{<t}\right) = \prod_{i=1}^{N} p\left(x_{t,i} \mid \mathbf{x}_{<t}\right). \tag{2}$$

In other words, a single neural network predicts the parameters of a multivariate conditional distribution with diagonal covariance, corresponding to the acoustic features of a single frame.

The main reason for choosing this model is that, unlike raw audio waveform, features produced by a parametric vocoder have two dimensions, similar to (single channel) images. However, unlike images, these two dimensions are not both spatial dimensions, but rather time-frequency dimensions. The translation invariance that 2D convolutions offer is an undesirable property for the frequency (or cepstral *quefrency*) dimension. Therefore, we model the features as 1D data with multiple channels. Note that these channels are only independent within the current frame; the prediction of each of the features in the current frame still depends on *all* of the features of *all* past frames within the receptive field (the range of input samples that affect a single output sample). This can be explained easily as all input channels of the initial causal convolution contribute to all resulting feature maps, and so on for the other convolutions.

Predicting all channels at once rather than one-by-one simplifies the models as it avoids the need for masking channels and separating them in groups. This approach is similar to [7] where all three RGB channels of a pixel in an image are predicted at once, although in our work we do not incorporate additional linear dependencies between channel means.

The network we propose, depicted in fig. 1, shares most of its architecture with WaveNet. Like this model we use gated convolutional units instead of gated recurrent units such as LSTM to speed up training. The input is fed through an initial causal convolution which is then followed by stacks of 2×1 dilated convolutions [21] where the dilation factor is doubled for each layer. This allows exponentially growing the model's receptive field, while linearly increasing the number of required parameters. To increase the total non-linearity of the model without excessively growing its receptive field, the dilation factor is increased up to a limit and then the sequence is repeated. We use residual and skip connections to facilitate training deeper networks [22]. As we wish to control the synthesizer by inputting lyrics, we use a conditional version of the model. At every layer, before the gated non-linearity, feature maps derived from linguistic features are summed to the feature maps from the layer's main convolution. In our case we do the same thing at the output stack, similar to [6].

3.1. Multi-stream architecture

Most parametric vocoders separate the speech signal into several components. In our case we obtain three *feature streams*; a harmonic spectral envelope, an aperiodicity envelope and a voiced/unvoiced decision (continuous pitch is given as a control input). These components are largely independent, but their coherence is important (e.g. synthesizing a harmonic component corresponding to a voiced frame as unvoiced will generally cause artifacts). Rather than jointly modeling all data streams with a single model, we decided to model these components using independent networks. This allows us to use different architectures for each stream and, more importantly, avoids one stream possibly interfering with the other streams. For instance, the harmonic component is by far the most important, therefore we wouldn't want any other jointly modeled stream potentially reducing model capacity dedicated to this component.

To encourage predictions to be coherent, we use predictions of one network as the input of another. We currently condition the voiced/unvoiced decision on the harmonic component and the aperiodic component on both harmonic component and voiced/unvoiced decision. All the networks are similar, but can

have different hyper-parameters (e.g. receptive field, early stopping). The voiced/unvoiced decision network has a Bernoulli output distribution rather than a mixture density (see 3.2). As we found this architecture to be satisfactory, we did not investigate the many other possible variations.

3.2. Constrained mixture density output

Many of the architectures on which we base our model predict categorical distributions, using a softmax output. The advantage of this approach is that no a priori assumptions have to be made about the (conditional) distribution of the data, allowing things such as skewed or truncated distributions, multiple modes, and so on. Drawbacks of this approach include an increase in model parameters, values are no longer ordinal, and the need to discretize data which is not naturally discrete or has high bitdepth.

Because our model predicts an entire frame at once, the issue of increased parameter count is aggravated. Instead, we opted to use a mixture density output similar to [7]. This decision was partially motivated because in earlier versions of our model with softmax output [23], we noted the predicted distributions were generally quite close to Gaussian or skewed Gaussian. In our model we use a mixture of four continuous Gaussian components, constrained in such a way that there are only four free parameters (mean, variance, skewness and a shape parameter). We found such constraints to be useful to avoid certain pathological distributions, and in our case explicitly not allowing multi-modal distributions was helpful to improve results. We also found this approach to speed up convergence compared to using categorical output.

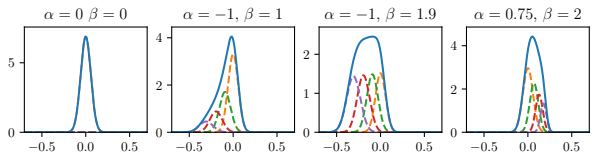

Figure 2: *Example distributions of the constrained mixture density output. All subplots use* $\mu = 0$ *and* $\sigma = 0.06$*, but varying skewness* α *and shape* β*. Y-axis scaled by* 10^3*.*

3.3. Robust generation by regularization

One of the principal issues during training was that the log likelihood of the training or validation set is often not very indicative of the final synthesis quality. The most prominent symptom of this issue is that the model may occasionally produce phonemes different from those given in its control input.

One reason for this may be that the training objective does not exactly match the generation setting. During training many samples are predicted in parallel, conditioned on actual past observations. However, in generation, samples are generated one-by-one in sequential order, each one conditioned on past predictions rather than past observations. Thus the model may overfit to observations from the dataset, causing generation to fail whenever predictions diverge even slightly. We expect that this issue is more noticeable in our case because we use relatively small datasets. Additionally, the data from a parametric vocoder is inherently less structured than other types of data such as raw waveforms or natural images, that is, it tends to be smoothly varying in time, have low amounts of noise, etc.

In order to make the generation process more robust to prediction errors, we propose to use a denoising objective,

$$\mathcal{L} = -\,\mathbb{E}_{\mathbf{x}_t}\,\mathbb{E}_{\tilde{\mathbf{x}}_{<t}\sim p(\tilde{\mathbf{x}}_{<t}|\mathbf{x}_{<t})}[\log p(\mathbf{x}_i|\tilde{\mathbf{x}}_{<t})], \tag{3}$$

where $p(\tilde{\mathbf{x}}|\mathbf{x})$ is a Gaussian corruption distribution,

$$p(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \lambda I), \tag{4}$$

with noise level $\lambda \geq 0$. That is, Gaussian noise is added to the input of the network, while the network is trained to predict the uncorrupted target.

When sufficiently large values of λ are used, this technique is very effective for solving the issue of overfitting. However, the generated output can also become noticeably more noisy. One way to reduce this undesirable side effect is to apply some post processing to the predicted output distribution, much in the same vein as the *temperature softmax* used in similar models, e.g. [6]. Another way to view this is that as temperature goes towards zero, the parameter generation criteria goes towards maximum likelihood, which is a commonly used in TTS systems.

We have also tried other regularization techniques, such as *drop-out*, but found them to be ultimately inferior to simply injecting input noise. One possible explanation for this is that adding noise to observed context is similar to the noise introduced by prediction errors during autoregressive generation.

3.4. Fast generation on CPU

One drawback of autoregressive models such as the proposed system is that generation is inherently sequential and thus cannot exploit massively parallel hardware such as modern GPUs. Naive implementations of the generation algorithm thus tend to be much slower than for many other types of models. We have independently developed a fast generation algorithm using caching techniques similar to those used in [24, 12]. Combining this with our frame-wise model, we can achieve speeds of 20-35$\times$ real-time on CPU. These runtimes, and the low memory and disk footprint make our system competitive with most existing systems in terms of deployability.

4. Experimental conditions

4.1. Acoustic and linguistic front-end

We use an acoustic front-end based on the WORLD vocoder [25] with a 32 kHz sample rate and 5 ms hop time. The dimensionality of the harmonic component is reduced to 60 coefficients by truncated frequency warping in the cepstral domain [26], using an all-pole warping coefficient $\alpha = 0.45$. To facilitate interpretation, the coefficients are finally converted back to frequency warped log-spectral features. The dimensionality of the aperiodic component is reduced to 4 coefficients by exploiting WORLD's inherently band-wise aperiodic analysis.

The linguistic features we use are relatively simple compared to most TTS systems as we do not model prosody. We use previous, current and next phoneme identity as one-hot encoded vectors. Additionally, we include the normalized position of the current frame within the current phoneme as a 3-state coarse coded vector, roughly corresponding to the probability of being in the beginning, middle or end of the phoneme. We do not use any features related to duration as our datasets have relatively uniform phoneme durations which may vary significantly from synthesis durations (esp. for vowels). The linguistic features are aligned to the acoustic features using a speaker-dependent Hidden Semi-Markov Model (HSMM) trained using deterministic annealing [27].

4.2. Datasets

In the initial evaluation of our system, we use three voices; English male and female voices (M1, F1), and a Spanish female voice (F2). The recordings consist of short sentences which were sung at a single pitch and an approximately constant cadence. The sentences were selected to favor high diphone coverage. The Spanish dataset contains 123 sentences, while the English datasets contain 524 sentences (approx. 16 and 35 minutes respectively, including silences). Note that these datasets are small compared to the datasets typically used to train TTS systems, but this is a realistic constraint given the difficulty and cost of recording a professional singer.

4.3. Configuration and hyper-parameters

We use an initial causal convolution operating on 10 past values, followed by a stack of 2×1 convolutions with dilation factors [1, 2, 4, 1, 2]. This results in a total receptive field of $105\,\mathrm{ms}$. For the harmonic feature stream, we use 100 channels in the convolutions, 240 channels for the skip connections. For the aperiodicity feature stream, we use 20 channels in the convolutions, 20 channels for the skip connections. The voiced/unvoiced decision stream uses 20 channels in the convolutions and 4 channels in the skip connections. All networks use a single output stage with $\tanh$ non-linearity. We use mini-batches of 16 sequences, each with an output length of 210 frames. We use the Adam optimizer [28] with an initial learning rate of 5×10^{-4}. The model was trained for a total of 1000 epochs, taking around 8 hours on a single Titan X Pascal GPU. The entire multi-stream network contains approx. 747k trainable parameters. While we found these settings to work well experimentally, they have not been exhaustively optimized.

5. Evaluation

We compare our system ("NPSS") against two alternative systems. The first is a HMM-based system ("HTS") build using the HTS toolkit (version 2.3) [29]. Mostly standard settings were used, except for a somewhat simplified context dependency (just the two previous and two following phonemes). The second system ("IS16") [1] is based on concatenative synthesis and was the highest rated system in the Interspeech 2016 Singing Synthesis Challenge.

In table 1, we show some quantitative results comparing there systems to our proposed method. These metrics were compute over a 10% validation split, silence frames and frames with mismatched voiced/unvoiced decision between target and prediction were excluded. The IS16 system is omitted from this table because the low redundancy of diphones in the dataset would force this system to find replacements for some diphones in the held-out validation utterances, giving the system an unfair disadvantage.

After the quantitative experiments we re-trained our models using the full datasets and synthesized one song for each of the three voices. To be able to better compare the different systems, we used pitch and phonetic timings from target recordings. From each song we extracted two short (under 10s) excerpts and made versions with and without background music. These stimuli were presented in 24 pairs to 18 participants who rated their preference between the different systems. The results of this listening tests are summarized in figure 3. Full versions of the songs used in the listening test are available at: http://www.dtic.upf.edu/~mblaauw/IS2017_NPSS/

Table 1: *Quantitative results for each of the voices. The first two columns show average Mel-Cepstral Distortion (MCD) for harmonic and aperiodic components respectively. The final column shows the accuracy of the voiced/unvoiced decision prediction.*

	Harm. (dB)	Aper. (dB)	V/UV (acc.)
HTS (M1)	4.24	0.89	96.86
NPSS* (M1)	4.40	1.02	97.39
HTS (F1)	4.28	1.50	96.93
NPSS* (F1)	4.29	1.62	97.54
HTS (F2)	4.21	1.27	98.40
NPSS* (F2)	4.43	1.47	98.51

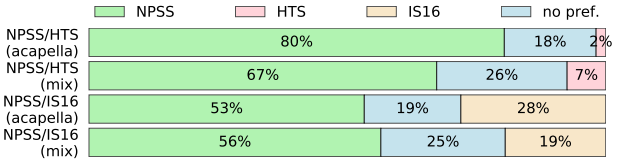

Figure 3: *Results of the preference test we used to qualitatively compare our model ("NPSS") to two other approaches ("HTS" and "IS16"), with and without background music (acapella).*

6. Conclusions

We presented a singing synthesizer based on neural networks that can be successfully trained on relatively small amounts of data. Listening tests showed a notable preference for our system compared to a statistical parametric system, and a moderate preference compared to a concatenative system. Interestingly, our quantitative metrics do not reflect this, showing nearly identical results. On the one hand this could be explained because the frame-wise metrics do not penalize trajectories that are overly smooth in time, yet this is perceptually important. On the other hand, these metrics do penalize small, perceptually irrelevant time-misalignments between predictions and targets, which autoregressive models may be more prone to produce. Compared to the HMM-based baseline system, we consider our method to sound noticeably less static and "buzzy", reproducing consonants more faithfully, and producing more natural sustained vowels. Compared to the concatenative system, we feel our method produces a similar overall sound quality. However, in certain segments, such as fast singing, subtle errors in segmentation become evident in the concatenative system. Generating phonetic contexts not in the training set is also generally handled better. The fast CPU-based autoregressive generation algorithm allows for many practical applications of our system. We hope that in the near future the flexibility offered by neural network can be explored further. In particular the area of multi-speaker training is promising, as it might help to overcome the issue of limited dataset sizes typical of singing voice.

7. Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research. We also thank Zya for providing the English datasets. Voctro Labs provided the Spanish dataset and the implementation of the fast generation algorithm. This work is partially supported by the Spanish Ministry of Economy and Competitiveness under the CASAS project (TIN2015-70816-R).

8. References

[1] J. Bonada, M. Umbert, and M. Blaauw, "Expressive singing synthesis based on unit selection for the singing synthesis challenge 2016," in *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12*, 2016, pp. 1230–1234.

[2] K. Saino, H. Zen, Y. Nankaku, A. Lee, and K. Tokuda, "An HMM-based singing voice synthesis system," in *Interspeech 2006 - ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17-21*, 2006.

[3] K. Oura, A. Mase, T. Yamada, S. Muto, Y. Nankaku, and K. Tokuda, "Recent development of the HMM-based singing voice synthesis system - Sinsy," in *The Seventh ISCA Tutorial and Research Workshop on Speech Synthesis, Kyoto, Japan, September 22-24*, 2010, pp. 211–216.

[4] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016.

[5] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with PixelCNN decoders," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 4790–4798.

[6] S. Reed, A. van den Oord, N. Kalchbrenner, V. Bapst, M. Botvinick, and N. de Freitas, "Generating interpretable images with controllable structure," Google DeepMind, Tech. Rep., 2016.

[7] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications," in *Proceedings of the International Conference on Learning Representations (ICLR-17)*, 2017.

[8] N. Kalchbrenner, A. van den Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu, "Video pixel networks," *CoRR*, vol. abs/1610.00527, 2016.

[9] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *CoRR*, vol. abs/1610.10099, 2016.

[10] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," in *Proceedings of the International Conference on Learning Representations (ICLR-17)*, 2017.

[11] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, "Char2Wav: End-to-end speech synthesis," in *Proceedings of the International Conference on Learning Representations (ICLR-17)*, 2017.

[12] S. O. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, J. Raiman, S. Sengupta, and M. Shoeybi, "Deep Voice: Real-time neural text-to-speech," in *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*, 2017.

[13] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014, pp. 3872–3876.

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

[15] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-00)*, vol. 3, 2000, pp. 1315–1318.

[16] J. Bonada and X. Serra, "Synthesis of the singing voice by performance sampling and spectral models," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 67–79, March 2007.

[17] A. Mase, K. Oura, Y. Nankaku, and K. Tokuda, "HMM-based singing voice synthesis system using pitch-shifted pseudo training data," in *Interspeech 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30*, 2010, pp. 845–848.

[18] K. Oura, A. Mase, Y. Nankaku, and K. Tokuda, "Pitch adaptive training for HMM-based singing voice synthesis," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30*, 2012, pp. 5377–5380.

[19] K. Shirota, K. Nakamura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "Integration of speaker and pitch adaptive training for HMM-based singing voice synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9*, 2014, pp. 2559–2563.

[20] M. Nishimura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "Singing voice synthesis based on deep neural networks," in *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12*, 2016, pp. 2478–2482.

[21] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proceedings of the International Conference on Learning Representations (ICLR-16)*, 2016.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30*, 2016, pp. 770–778.

[23] M. Blaauw and J. Bonada, "A singing synthesizer based on PixelCNN," http://www.dtic.upf.edu/~mblaauw/MdM_NIPS_seminar/, 2016.

[24] P. Ramachandran, T. L. Paine, P. Khorrami, M. Babaeizadeh, S. Chang, Y. Zhang, M. Hasegawa-Johnson, R. Campbell, and T. Huang, "Fast generation for convolutional autoregressive models," in *Proceedings of the International Conference on Learning Representations (ICLR-17)*, 2017.

[25] 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*, vol. 99, pp. 1877–1884, 2016.

[26] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai, "Mel-generalized cepstral analysis - a unified approach to speech spectral estimation," in *Proceedings of the International Conference on Spoken Language Processing (ICSLP-94)*, 1994.

[27] N. Ueda and R. Nakano, "Deterministic annealing EM algorithm," *Neural networks*, vol. 11, no. 2, pp. 271–282, 1998.

[28] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR-15)*, 2015.

[29] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, "The HMM-based speech synthesis system (HTS) version 2.0," in *Sixth ISCA Workshop on Speech Synthesis (SSW6), Bonn, Germany, August 22-24*, 2007, pp. 294–299.