SINGING SYNTHESIS: WITH A LITTLE HELP FROM MY ATTENTION.

Orazio Angelini   *Alexis Moinet*   *Kayoko Yanagisawa*   *Thomas Drugman*

Amazon Research, Cambridge

`{aorazio, amoinet, yakayoko, drugman}@amazon.com`

Abstract

We present UTACO, a singing synthesis model based on an attention-based sequence-to-sequence mechanism and a vocoder based on dilated causal convolutions. These two classes of models have significantly affected the field of text-to-speech, but have never been thoroughly applied to the task of singing synthesis. UTACO demonstrates that attention can be successfully applied to the singing synthesis field and improves naturalness over the state of the art. The system requires considerably less explicit modelling of voice features such as F0 patterns, vibratos, and note and phoneme durations, than previous models in the literature. Despite this, it shows a strong improvement in naturalness with respect to previous neural singing synthesis models. The model does not require any durations or pitch patterns as inputs, and learns to insert vibrato autonomously according to the musical context. However, we observe that, by completely dispensing with any explicit duration modelling it becomes harder to obtain the fine control of timing needed to exactly match the tempo of a song.

Index Terms: Singing voice synthesis, attention, machine learning, deep learning, autoregressive models

1. Introduction

Research efforts on computer-aided singing synthesis systems date back to the late 1950s [1]. Historically, the working principles of singing synthesis systems have been based on parametric text-to-speech (TTS) or unit selection technology. Notable recent examples are Sinsy [2], a statistical parametric singing synthesis system, and Vocaloid [3], based on unit selection. A recent development in the field is the introduction of deep neural networks (DNN) [4]. The latest version of Sinsy adopts DNNs [5], and DNN sub-models exist to predict specific features of a singing voice, such as F0 [6] or note transition and sustain patterns [7]. The introduction of the WaveNet architecture [8] marked an increase in the importance of DNN techniques for TTS. Singing synthesis followed, with the introduction of several DNN-based models. Examples include [9, 10, 11], which present variations on a model of a singing voice based on WaveNet[8], and [12] developing a WGAN architecture. A common feature of all these singing synthesis models is the need to develop a number of separate specialised sub-models to predict specific voice features such as the F0 contour, the duration of individual phonemes, or the start time of notes (which, in natural singing voices, do not follow exactly the timing of the score [2]). A development in TTS technology that is relevant to our work has been the introduction of attention-based architectures [13] such as Tacotron [14, 15] and Deep Voice [16], attention-based sequence-to-sequence (AS2S) models which predict spectrograms that are subsequently used to synthesise a waveform with a vocoder. For our purposes, the most salient feature of AS2S architectures is that its only conditioning input is text (or a corresponding phoneme list), and not any additional model or piece of information. Whereas previous

models needed to be conditioned on several other pieces of context, for example F0, an AS2S autonomously learns an implicit model of all voice features that are not included in its inputs: e.g. intonation, stress and rhythm. This point is made explicitly in [17, 18, 19], which try to learn an explicit representation for these features to be used in later conditioning. [20] even uses it in singing, with an attention model that is fed pitch and duration externally. In this paper, we consider the possibility that an AS2S architecture may be able to learn an implicit model of singing interpretation in a similar way to what it does for speech prosody. We train an AS2S architecture on singing data, and observe that it is capable of generalising to unseen musical scores. We find that AS2S architectures, conditioned only on the information available in a score, are capable of singing synthesis, and they can significantly improve naturalness with respect to the state of the art. We name our model UTACO, from the Japanese word "歌"("uta"), meaning song, and the beginning of the word "Tacotron".

2. System description

UTACO consists of three main parts (see Fig.1). The first is a frontend that receives a score in MusicXML (MXL) format [21] as input, and outputs the note embeddings to be sent to an attention encoder. The second is an AS2S architecture, based on [22, 23], modified to accept the note embeddings, whose decoder produces mel-spectrograms. The spectrograms are finally synthesized with a vocoder. The frontend performs linguistic analysis on the score lyrics. The phoneme sequence for the utterance is inferred from the lyrics text. We allow for 3 possible vowel levels of stress (0,1,2). The punctuation is ignored. Then the frontend determines which phonemes correspond to each note of the score, using syllabification information specified in the MusicXML file. It also computes the duration in seconds of each note according to the score, and finally combines this information into embeddings that will be used to condition the AS2S model. The modification applied to the TTS AS2S architecture concerns the conditioning inputs to the encoder. Whereas a TTS AS2S generally takes as its only input a sequence of (one-hot encoded) tokens representing phoneme IDs for the utterance to be generated, our system uses phoneme-level note embeddings. These consist of 5 streams, all of equal length and concatenated:

1. The phoneme sequence for the song utterance to be generated, one-hot encoded. 84 tokens are available in this stream, including a start (`<s>`), and a word boundary (`<wb>`) one.

2. The octave sequence for the note to be sung on each phoneme, according to the score, one-hot encoded. For example, for the sequence of notes C4, D#4, G3, the corresponding octave sequence would be (4, 4, 3). We allow for 4 values, which is the range found in our dataset.

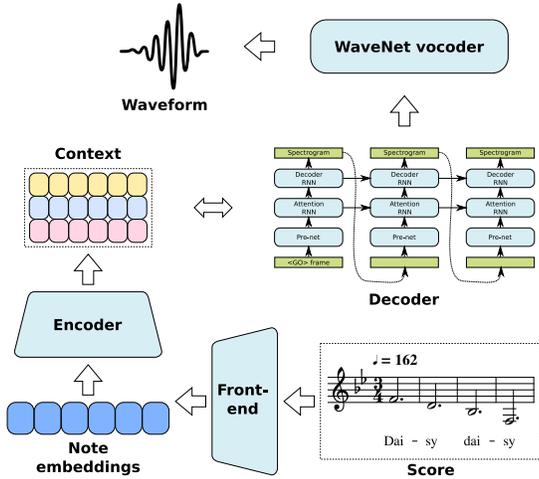3. The step in the chromatic scale (out of 12 possible ones)
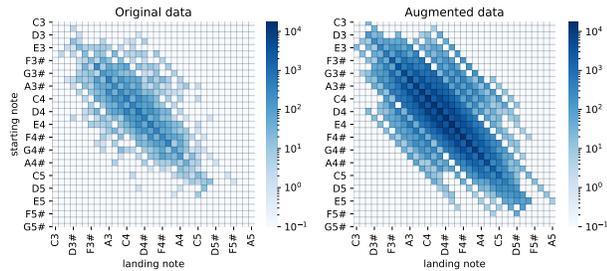
Figure 1: *Diagram of UTACO's arhitecture.*

Figure 2: *Distribution of pitch changes in the dataset. Left (right) panel shows the distribution of the original (of augmented) data.*
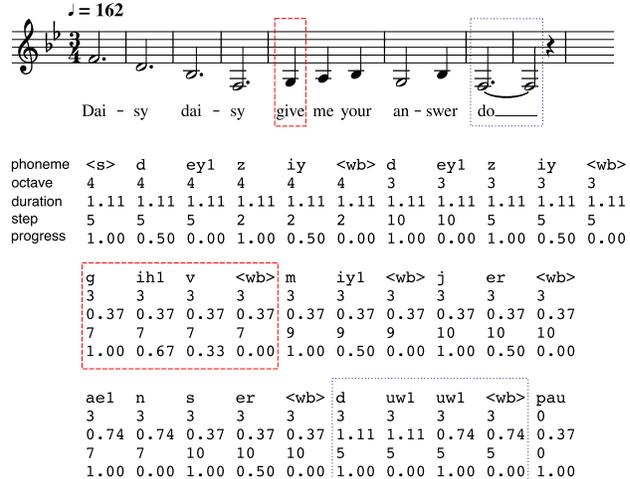

Figure 3: *An example of the embeddings detailed in Sec.2. Dashed red highlights the example illustrated in Sec.2, and dotted blue shows a tie disambiguated by stream 5.*

Rests are represented by a pause token, and an additional one-hot value for octave and step. An example of the embeddings produced can be seen in Figure 3. Compared to analogous solutions, we avoided the use of embeddings in the style of [24], which are more useful in polyphonic music. Our inputs are less rich compared to [10], which will be discussed in Sec.5. The rest of the architecture is an AS2S model based on [23]. It was trained with the Adam optimisation algorithm [25] and a learning rate of 0.001 for ∼300K steps. The network so organised is capable, after standard training on a large enough dataset, to produce spectrograms which are then synthesised by a vocoder, see Sec.3.1 and 3.3 for more detail.

3. Experimental protocol

3.1. Dataset

The dataset consists of 96 songs in US English, sung a cappella by a single female voice, for a total of 2 hours and 15 seconds of music. It covers several genres, such as pop, blues rock, and some children's songs. The songs have been autotuned to correct the performer's pitch errors. Since the length of most songs is in the order of minutes, we split them into segments of ∼20-30 seconds, which correspond to ∼200 phonemes. This reduces the memory requirements at training time compared to processing whole songs – due to the attention matrix size increasing with the square of the sequence length – while keeping the batch size large enough, at 32. In the vocoder training set we used an additional ∼40 hours of speech data by the same speaker performing the songs. Given the small amount of data available, we reduced the test set size to the minimum possible, of about 5 minutes in total. Most songs contain repetitions such as refrains or repeated pitch patterns. Holding out whole songs would have been too costly in terms of training data, and yielding too little diversification in the test set. In order to ensure a strict separation of train and test data, we compared all possible pairs of segments. If two segments included a sequence of 3 or more identical subsequent note pitches in the score, we excluded both from the test set. Apart from the segments excluded because of repetitions, we selected the test set segments randomly.

for the note to be sung on each phoneme, according to the score, e.g. G#, or B-, one-hot encoded.

4. The duration in seconds of the note to be sung on each phoneme, represented as a floating point number (z-score normalised).

5. A position embedding computed as a ramp representing the advancement of the note for each phoneme that it contains, as a floating point number in the interval [0,1]. For example, if three phonemes have to be sung on a given note, the first phoneme will have 1.0 on this stream, the second 0.5, and the last 0.0.

Streams 2-4 are repeated for the length of the note. For example, if the word "give" is to be sung on a G3 for 0.37 seconds, the tokens ("g", "ih1", "v", "<wb>") will be put in stream 1 (as in Fig.3), stream 2 will contain the octave (3, 3, 3, 3), the corresponding positions in stream 3 will contain (G, G, G, G), and those in stream 4 will contain the z-score normalised values corresponding to (0.37, 0.37, 0.37, 0.37). Stream 5 will contain (1.0, 0.67, 0.33, 0.0). If a phoneme has to be sung on several notes, the phoneme is repeated while the other streams change. For example, to sing the phoneme "uw1" on two notes, G and E, each of duration 0.1 seconds, followed by one 0.2 second F, then stream 1 will contain ("uw1", "uw1", "uw1"); while stream 3 will contain (G, E, F) and stream 4 (0.1, 0.1, 0.2). The ramp also helps disambiguate ties, such as the one in Fig.3.

3.2. Data augmentation

A common concern in the singing synthesis field consists of the small amount of data available to train models [10]. Song data is symmetric to two transformations: pitch shifting and tempo changes. Following previous literature [10, 26, 5], we employed both. We applied the following transformations to each song in the dataset:

- Shifting the pitch by [-1, 0, +1, +2, +3] semitones.

- Changing the original beats per minute (bpm) of the song to [.85, .90, .95, 1., 1.05, 1.10, 1.15] percent of the original one.

There are 35 possible combinations of these two augmentation types, so the final amount of augmented data consisted of about 70 hours. This makes more contexts available to the model during training. The change can be visualised in Figure 2. We applied these transformations using an algorithm that preserves perceived vocal tract length. The maximum amount of change that can be applied before excessive degradation has been determined through informal listening tests by the authors. Despite our attempts, the model without augmentations was unstable and produced mediocre quality when inferring on training data, and produced random phonemes and melody at inference on unseen inputs. We believe that this is because the small number of contexts available in the unaugmented datasets makes the model unable to extrapolate to unseen data. Therefore, we did not include the model trained on unaugmented data in our experimental validation.

3.3. Vocoder

The vocoder used in this paper is an autoregressive WaveNet based on [8], conditioned only on (80-dimensional) mel-spectrograms as in [15].

The training data for the vocoder used in our test consisted of the whole training set (including augmentations) used for the AS2S model plus ∼40 hours of speech data by the same speaker. We observed that the addition of speech data to the vocoder training set seems to increase the quality of the samples.

3.4. Baseline

We compared UTACO to WGANSing [12], the only recent state-of-the-art (SOTA) neural singing synthesis model for which a training protocol was released. The system is inspired by Deep Convolutional Generative Adversarial Networks (DCGAN), with a Wasserstein-GAN loss, and uses the WORLD vocoder [27]. We compared our proposed system with WGANSing trained on the same (augmented) dataset as our model, using a slightly modified version of the protocol in the published repository [1]. Specifically, the corpus used in [12] includes both a sung and spoken version of the same text for each utterance. In the published protocol, both are used for training. Since our dataset does not have the necessary spoken data, we only trained WGANSing on song data. The effect of this change is discussed in Sec.3.5.1. Note that WGANSing needs to be fed external duration and continuous pitch, which we extracted from the original recordings. This needs to be kept in mind for a fair comparison, since UTACO generates its own timing and pitch patterns.

3.5. MUSHRA methodology

We set up MUSHRA tests [28] comparing various versions of each segment sung or synthesised in the same voice. As recommended in [28], the segments chosen for the test set were further split into chunks, each ≤10 seconds long. Cuts were made in naturally occurring pauses in the songs, trying to keep the segments as long as possible. This generated typical lengths of ∼3-8 seconds. Each segment was judged by US English native speakers, who were asked a question with this exact wording: "Rate the samples in terms of their naturalness". The interpretation of the word "naturalness" is left to the listeners. They rated the samples between 0 (representing "Not at all natural") and 100 (representing "Completely natural"). In all experiments, all possible combinations of pairwise 2-sided t-tests on the means of the scores for different systems yield a p-value ≪0.001.

3.5.1. MUSHRA for baseline validation

We checked that training WGANSing without any spoken data does not alter the model performance and is therefore representative of the model's quality. We trained WGANSing on the same corpus[2] used by the authors in their publication [12]. We compared, in a separate MUSHRA test, their published audio clips[3] to the ones we produced. To replicate our MUSHRA methodology, the audio clips have been cut into 51 segments. 30 listeners were involved in the evaluation, and each of them listened to all of the segments. We trained a model with speech data and another one without. The results are shown in Fig.4. Surprisingly, WGANSing with no speech in the training data (`no_speech`) has a better score than the model with speech (`with_speech`). The difference in mean MUSHRA score between our best model (`no_speech`) and the results published by the authors (`published`) is 5.89. The two score distributions for `no_speech` and `published` are very similar, except for slightly fatter tails on the lower and higher end of the score spectrum, respectively. Therefore, we conclude that training WGANSing on singing data alone does not deteriorate the model's overall performance.

3.5.2. Mushra for proposed model

This MUSHRA test asked 40 listeners to compare 3 versions of each of 74 segments. Each listener examined all of the segments. The upper anchor is an original recording, compared to the same segment synthesised by UTACO. The lower anchor is WGANSing, representative of SOTA, described in Sec.3.4. Note that the MUSHRA methodology requires that we train the models compared on the same data, in order to remove dataset bias. Because of this, WGANSing was trained on the same corpus we used for the proposed system.

4. Results

The MUSHRA test results, presented in Figure 5 show that the mean relative MUSHRA score for UTACO is 60.45, compared to recordings which have 81.62. Compared to the baseline, at 30.82, there is a significant improvement in naturalness over the SOTA. Upon closer inspection, we observed that most of UTACO's segments in the lower score quartile (whose scores range between 26-50) contain either a vocoder glitch (there are

---

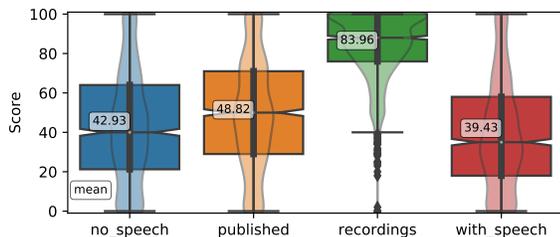Figure 4: *Box and violin plot of MUSHRA for the baseline, trained on WGANSing's corpus.*

Figure 5: *Box and violin plot of MUSHRA for UTACO, trained on our corpus.*

3 in the mushra dataset) or mumbled/mispronounced words. Segments in the upper two quartiles, which obtain average scores in the range 61-77, seem to show much less of these problems[4]. The main improvement might be due to the clarity and expressivity of our model compared to the baseline model. The MUSHRA listeners commented on the sound quality of the baseline defining it "muffled", "glitchy", "poor". Regarding the naturalness of the voice, the baseline was perceived as "monotonous" and "too regular to sound natural", although at inference time it was fed oracle F0 and phoneme timings from the original recordings. In contrast to [10], UTACO sings in tune autonomously. We found that training on non-autotuned data produces a model that often goes out of tune. UTACO performs best on simpler songs, which do not include very high or low-pitched notes, or phonemes sustained for a long time. Thich was expected given these contexts are under-represented in the data (see Fig.2). Our model suffers from some articulation difficulties on low-frequency phoneme combinations. MUSHRA listeners noticed this, and commented about "mispronunciations", but this was expected since we used only ~2 hours of unaugmented data. Since the dataset did not contain vibrato annotations, no explicit indication of it was included in the note embeddings. Nevertheless, we observed that the model learns to reproduce a good vibrato, and apply it in the right places – on longer sustained notes – according to the musical context. The two main drawbacks of UTACO are due to the nature of the architecture of choice. First, we verified that when a long rest, i.e. silence, is encountered in the score, the duration of the silence produced by the encoder is completely unpredictable. We observed that the concentration of the attention weights decreases significantly on pause tokens, leading to instability. This issue, similar to word-skipping, seems to be common to AS2S models in general [15, 16], but is more severe in music, where pauses of any given length are frequent and essential to the musical context and rhythm. Another possibly related drawback is that UTACO seems to produce notes that are slightly too long or too short, losing the rhythm over time. This could be due to the combined effect of lack of data and augmentations. Changing the bpm of a song affects all durations uniformly, so any discrepancies between score duration and phoneme duration in the training data are correlated among augmented versions of the same song, which would essentially lead to overfitting on errors. In our architecture it is not possible to directly control the timing of the attention matrix. None of the two problems reduce its ultimate ability to synthesise a singing voice. Two easy workarounds are cutting the scores
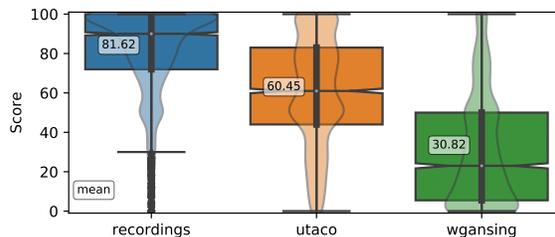
on rests, and editing the tempo in post-processing in case the timing problem manifests itself. We observed that low-scored samples from the MUSHRA test seem to suffer more frequently from vocoder glitches and mumbling than attention instabilities. We attribute the former to the lack of data.

5. Discussion

In this paper we presented UTACO, to the best of our knowledge, the first use of the attention mechanism to generate pitch and timing in the field of singing synthesis. What sets the system apart from other techniques, other than the noticeable improvement in naturalness over the SOTA, is the complete lack of need for explicitly modelling many parts of the song synthesis process. The AS2S architecture is capable of autonomously modelling F0 patterns, vibratos, and inserting vibrato in the right context. Training UTACO requires a dataset whose size is the same order of magnitude as typical datasets in the field [4]. Its only input is the musical score with lyrics to be synthesised: it requires no explicit modelling of any feature of a singing voice. As noted earlier, our embeddings are leaner than those in [10], which also embed the previous/next phoneme, as well as other linguistic features and more importantly phoneme length. We hypothesise that the LSTM in the encoder plus the attention makes them redundant. Many of the previous singing synthesis systems require a separate model for allocating phoneme duration inside of a note. To increase naturalness, a model is sometimes needed to emulate the small imperfections in timing found in natural singing, such as in [2]. In other words, many duration values needed for the synthesis are not unambiguously specified by the score. Attention dispenses with the need to model duration, and is conceptually simpler than previous systems proposed in the literature. Therefore, one major improvement is the reduced amount of modelling work needed to create a singing synthesis system with the method we described here. However, the main drawbacks of employing an AS2S architecture are in the precision of timing, and are also a direct result of the attention model used. Some previous work (e.g. [16, 29]) already tackled the same problems. We propose that UTACO can be stabilised with further work, and that the benefits of a stable attention model would ultimately justify its use. AS2S models enjoy much active research work, and all new extensions of such architectures, such as speaker identity, language conditioning and style conditioning can potentially be applied to UTACO immediately.

6. References

[1] P. R. Cook, "Singing voice synthesis: History, current work, and future directions," *Computer Music Journal*, vol. 20, no. 3,

---

[4]Model samples are available at this address: https://www.amazon.science/blog/a-simpler-singing-synthesis-system.

pp. 38–46, 1996. [Online]. Available: http://www.jstor.org/stable/3680822

[2] Y. Hono, S. Murata, K. Nakamura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "Recent Development of the DNN-based Singing Voice Synthesis System-Sinsy," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). IEEE*, 2018.

[3] H. Kenmochi and Hayato Ohshita, "Vocaloid-commercial singing synthesizer based on sample concatenation." in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[4] E. Gomez, M. Blaauw, J. Bonada, P. Chandna, and H. Cuesta, "Deep Learning for Singing Processing: Achievements, Challenges and Impact on Singers and Listeners," *arXiv e-prints*, p. arXiv:1807.03046, Jul 2018.

[5] K. Nakamura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "Singing voice synthesis based on convolutional neural networks," *arXiv e-prints*, p. arXiv:1904.06868, Apr 2019.

[6] Y. Wada, R. Nishikimi, E. Nakamura, K. Itoyama, and K. Yoshii, "Sequential Generation of Singing F0 Contours from Musical Note Sequences Based on WaveNet," in *Sequential Generation of Singing F0 Contours from Musical Note Sequences Based on WaveNet*, 2018, pp. 983–989. [Online]. Available: http://eita-nakamura.github.io/articles/Wada_etal_SingingF0GenerationByWaveNet_2018.pdf

[7] K. Hua, "Modeling Singing F0 With Neural Network Driven Transition-Sustain Models," *arXiv e-prints*, p. arXiv:1803.04030, Mar 2018.

[8] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, 2016, pp. 125–125.

[9] M. Blaauw and J. Bonada, "A neural parametric singing synthesizer," 2017, pp. 4001–4005.

[10] ——, "A neural parametric singing synthesizer modeling timbre and expression from natural songs," *Applied Sciences*, vol. 7, no. 12, p. 1313, 2017.

[11] F. Bous and A. Roebel, "Analysing deep learning-spectral envelope prediction methods for singing synthesis," in *European Signal Processing Conference (EUSIPCO)*, 2019.

[12] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "WGansing: A multi-voice singing voice synthesizer based on the Wasserstein-Gan," in *European Signal Processing Conference (EUSIPCO)*, 2019.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[14] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *Proc. Interspeech 2017*, pp. 4006–4010, 2017.

[15] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.

[16] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning," *arXiv e-prints*, p. arXiv:1710.07654, Oct 2017.

[17] Y. Wang, D. Stanton, Y. Zhang, R.-S. Ryan, E. Battenberg, J. Shor, Y. Xiao, Y. Jia, F. Ren, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *International Conference on Machine Learning*, 2018, pp. 5180–5189.

[18] R. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. Weiss, R. Clark, and R. A. Saurous, "Towards end-to-end prosody transfer for expressive speech synthesis with tacotron," in *International Conference on Machine Learning*, 2018, pp. 4693–4702.

[19] V. Klimkov, S. Ronanki, J. Rohnke, and T. Drugman, "Fine-grained robust prosody transfer for single-speaker neural text-to-speech," *Proc. Interspeech 2019*, pp. 4440–4444, 2019.

[20] R. Valle, J. Li, R. Prenger, and B. Catanzaro, "Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens," *arXiv e-prints*, p. arXiv:1910.11997, Oct 2019.

[21] M. Good, "MusicXML in Commercial Applications | Songs and Schemas," *Music Analysis East and West*, pp. 9–20, 2006. [Online]. Available: https://michaelgood.info/publications/music/musicxml-in-commercial-applications/

[22] T. Merritt, B. Putrycz, A. Nadolski, T. Ye, D. Korzekwa, W. Dolecki, T. Drugman, V. Klimkov, A. Moinet, A. Breen *et al.*, "Comprehensive evaluation of statistical speech waveform synthesis," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 325–331.

[23] J. Latorre, J. Lachowicz, J. Lorenzo-Trueba, T. Merritt, T. Drugman, S. Ronanki, and V. Klimkov, "Effect of data reduction on sequence-to-sequence neural tts," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7075–7079.

[24] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "Music transformer: Generating music with long-term structure," *arXiv preprint arXiv:1809.04281*, 2018.

[25] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.

[26] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio Augmentation for Speech Recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[27] 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, no. 7, pp. 1877–1884, 2016.

[28] I. R. Assembly, "ITU-R BS. 1534-3: Method for the subjective assessment of intermediate quality level of audio systems. October 2015."

[29] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," in *Advances in Neural Information Processing Systems*, 2019, pp. 3165–3174.