SynthSing: A Singing Voice Synthesizer

James Bunning, Mu Yang, Sharada Murali, Shiyu Mou and Yixin Yang

September 30, 2018

1 Abstract

Recent advances in synthesizing the human singing voice from a music score and lyrics have been achieved using a Neural Parametric Singing Synthesizer (NPSS). This autoregressive model based on a modified version of the text-to-speech synthesis (TTS) WaveNet architecture trains on acoustic features extracted from a parametric vocoder to significantly reduce training and generation times while still achieving the sound quality and naturalness of state-of-the-art concatenative systems. Although the modified WaveNet architecture can model a specific singer's voice and a specific style of singing, it cannot generate a new singing performance given new speaker data. Our proposed system will expand on the neural parametric singing synthesizer to synthesize a singing voice in the style of a new singer.

2 Description of the system

2.1 Baseline Model

Our proposed system is based on an existing Neural Parametric Singing Synthesizer (NPSS) [1], which synthesizes singing by passing music notes and lyric phoneme sequences through three separately trained models: a phonetic timing model, a pitch model, and a timbre model, detailed in Figure 1.

During the first stage of generation, the phonetic timing model predicts the duration of each phoneme in the sequence to be synthesized from the musical note begin and end times and the phoneme sequence corresponding to each note (generally a syllable). Next, the pitch model predicts the fundamental frequency (F0) from the timed musical and phonetic information. The predicted sequence of timed phonemes and F0 are then fed as control inputs to the timbre model to generate the harmonic spectral envelope, aperiodicity envelope, and voiced/unvoiced (V/UV) decision required for synthesis. At the end of the chain, the synthesis stage of a parametric vocoder combines the estimated timbrical acoustic features with the predicted F0 to generate the singing voice waveform.

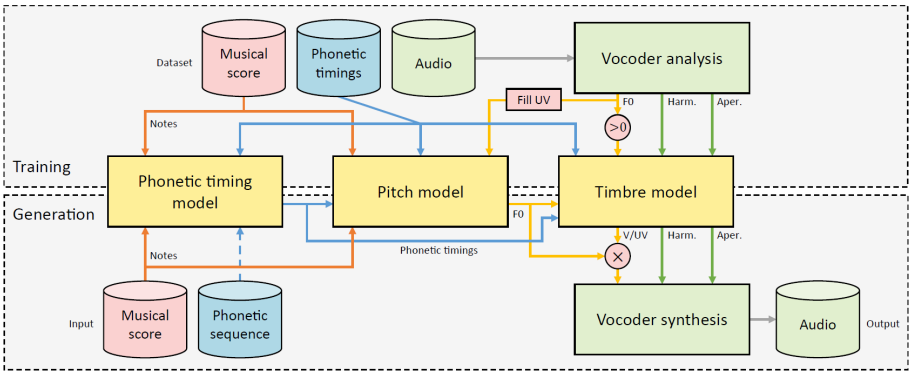

Figure 1: Overview of the NPSS system

Generation using the cascaded models is autoregressive, which means that each prediction depends on a window of past predictions.

The phonetic timing model is trained using a simple feedforward neural network, without the need for a recurrent or convolutional architecture. The inputs are timed-aligned music notes and phonetically-transcribed lyrics. To avoid making assumptions about the probability distribution of note onset deviations, a nonparametric approach is taken by using a softmax output and discretizing the deviations to multiples of the hoptime.

The pitch model and timbre model are both trained using a modified version of the WaveNet architecture[2]. Unlike WaveNet, which models the raw waveform, the NPSS system models features produced by a parametric vocoder, which separates the influence of pitch and timbre. This offers many advantages, including requiring fewer layers and model parameters, reducing training and generation times, and allowing training on smaller datasets. The one inherent disadvantage with using a vocoder is the degradation in sound quality; however, this effect is negligible when compared to degradations introduced by other current state-of-the-art singing synthesizer models.

The pitch model's modified WaveNet architecture predicts F0 from musical and phonetic control inputs. The true value of F0 is extracted from the raw waveform of an isolated singer using the WORLD parametric vocoder[3]. The F0 contour is modeled as-is, without any decomposition (e.g. separating vibratos or using source material without consonants). On the other hand, the timbre model consists of a multistream variant of the WaveNet architecture. The three feature streams used as input are the harmonic spectral envelope, aperiodicity envelope, and U/V decision. Similar to F0, these acoustic features are extracted using the WORLD parametric vocoder. The harmonic component is represented by 60 log

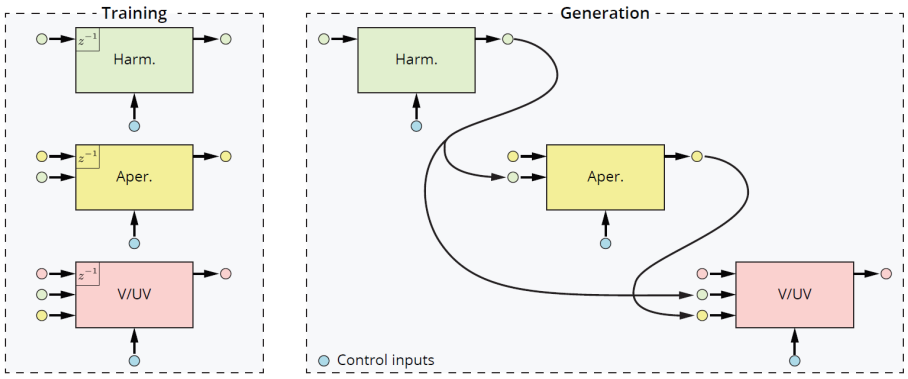

Figure 2: Cascaded multistream architecture of the timbre model

Mel-Frequency Spectral Coefficients (MFSCs), and the aperiodic component is represented by four aperiodic coefficients. Instead of jointly modeling all feature streams with a single model, each component is modeled using an independent network, shown in Figure 2.

This approach allows for more fine-tuned control over each stream's architecture and prevents streams with lower perceptual importance from interfering with streams of higher perceptual importance. For example, the harmonic component is by far the most perceptually significant feature, and therefore, the system should not allow other jointly modeled streams to reduce the model's capacity dedicated to this component. In order to produce coherent predictions, the outputs of each network are concatenated to the input of the next. That is, the aperiodic component depends on the harmonic component, and the U/V decision depends on both the harmonic and aperiodic components.

2.2 Transfer Learning for Multi-Speaker Singing Synthesis

A major feature of our proposed system will be generating singing in the voice of different speakers, including those not present during the training process. While there have been various such systems proposed for multispeaker speech synthesis[4, 5, 6], this problem has not been explored for singing synthesis.

In multispeaker TTS systems, synthesis can be done by training a speaker encoder network and neural vocoder network separately. Results have shown[4] that the speaker encoder and synthesis networks can be trained on unbalanced and disjoint datasets and still generalize very well.

In order for our system to synthesize a singing voice for new speakers, the network has to learn to decorrelate speaker-specific information from the singing

information. One way to do this is by padding the existing audio features with a one-hot vector containing information about the speaker identity. We also propose training an autoencoder to learn embeddings for singers, which are then conditioned upon during the training phase of the WaveNet framework. This autoencoder can also be integrated with the existing system and trained end-to-end.

Ultimately, our proposed model can be a prototype for a singing voice synthesizer that can use a specific singer's style and lyrics to generate a novel song. Using transfer learning, the system will be able to generate entire songs in a particular singer's style but with the voice of an entirely different speaker, using just a few sentences from the new speaker to learn speaker characteristics.

3 Description of Algorithms

3.1 Dataset Generation

Training the NPSS system requires a dataset of songs with phonetically transcribed and time-aligned lyrics. In order to obtain phonetic transcriptions, we will employ Gentle[7], a forced-aligner built on Kaldi. Forced aligner programs take audio files and transcripts and return extremely precise timing information for each phoneme in the media. Gentle constructs a language model composed of all bigrams within a provided transcript of lyrics and combines it with an acoustic model trained by Kaldi developers on the Fisher English corpus. After running an online transcription, the sequence alignment algorithms match the result with the original transcript.

3.2 WaveNet Algorithms

The pitch model and timbre model of the existing NPSS system are trained using a modified version of the WaveNet architecture. WaveNet is a fully probabilistic generative model, capable of generation conditioned on other control inputs. The structure of one layer is shown in Figure 3.

Some key features of WaveNet include:

- Dilated causal convolutions
  Causal convolution is an autoregressive approach, and dilated convolutions increase the size of receptive fields. The use of dilated causal convolutions enables WaveNet to predict new samples based on values in previous timestamps within a reasonably long range.

- Gated activation units

$$z = tanh(W_{f,k} * \mathbf{x}) \odot \sigma(W_{g,k} * \mathbf{x}) \tag{1}$$

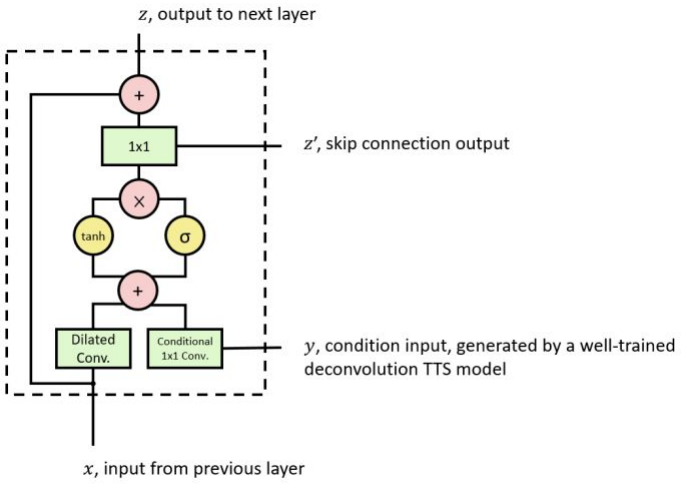

Figure 3: Structure of a single layer in WaveNet

Where k denotes layer index. f and g denote filter and gate respectively. $W_{f,*}$ is a learnable convolutional filter.

- Conditional input variables
  We have global conditioning (e.g. speaker representation) and local conditioning (e.g. linguistic features).
  For global conditioning:

$$z = tanh(W_{f,k} * \mathbf{x} + V_{f,k}^T \mathbf{h}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k}^T \mathbf{h}) \tag{2}$$

  Where $\mathbf{h}$ is a latent representation that influences the output across all timestamps, e.g. speaker identity. $V_{f,*}$, $V_{g,*}$ are learnable linear projections. Vector $V_{f,k}^T \mathbf{h}$ and $V_{g,k}^T \mathbf{h}$ are broadcast over time dimension.
  For local conditioning:
  Conditional vector $\mathbf{h}$ is a time series. It is first transformed to a new time series $\mathbf{y} = f(\mathbf{h})$, which can be a transposed convolutional network. It is then used in the activation unit as follows:

$$z = tanh(W_{f,k} * \mathbf{x} + V_{f,k} * \mathbf{y}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k} * \mathbf{y}) \tag{3}$$

  $V_{f,*}$ and $V_{g,*}$ are 1-by-1 convolution. Alternatively, $V_{f,*} * h$ can also be used directly and repeat the values accordingly over time. In this case, the $V_{f,*}$ and $V_{g,*}$ should also be a learnable linear projection.

- Residual and skip connections
  Residual and skip connections enable training of deeper networks.

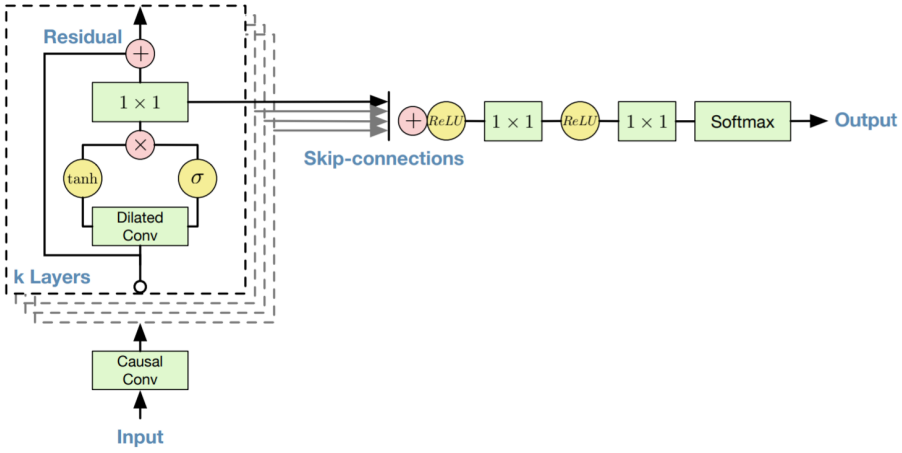

Figure 4: Overview of the complete WaveNet architecture

An overview of the complete WaveNet architecture is shown in Figure 4. Individual layers are stacked to form a deeper network. Skip connections and activation functions are used to compute the final output.

3.3 Neural Parametric Singing Synthesizer Algorithms

3.3.1 Pitch Model Algorithms

Data Augmentation Algorithm: Obtaining a dataset that sufficiently covers all notes in a singer's register can be challenging. Assuming that pitch gestures are largely independent of absolute pitch, the training data can be pitch shifted to ensure that all notes of a melody within a sample can occur at any note within the singer's register. While training, a pitch shift in semitones is drawn from a discrete uniform random distribution for each sample in the minibatch:

$$pshift \sim U(pshift_{min}, pshift_{max})$$

$$pshift_{min} = pshift_{min}^{singer} - pshift_{max}^{sample}$$

$$pshift_{max} = pshift_{max}^{singer} - pshift_{min}^{sample}$$

where $pshift_{min}$ and $pshift_{max}$ define the maximum range of pitch shift applied to each sample. This pitch shift is applied to both the pitch as a control input and the target output pitch:

$$\hat{pitch}_{cond} = pitch_{cond} + pshift$$

$$\hat{f}_0 = f_0 2^{\frac{1}{12} pshift}$$

Tuning Post-Processing Algorithm: Since the pitch model does not enforce the F0 contour to be in tune, the predicted pitch may sometimes deviate a certain amount from the note pitch. In expressive singing, it is not uncommon for F0 to stray from the note pitch for some notes and still sound perceptually in tune. The output generated by the pitch model can be improved by applying a tuning post-processing algorithm to the predicted pitch F0. For each note (or segment within a long note), the perceived pitch is estimated using F0 and its derivative. The smoothed difference between this pitch and the music note pitch is then used to correct the final pitch when generating the waveform.

3.3.2 Timbre Model Algorithms

Long Notes Algorithm: In most datasets, not all note durations will be exhaustively covered, which can result in problems when synthesizing notes significantly longer than the notes in the dataset. In order to reduce "stuttering" artifacts caused by time repetitions of transitions predicted by the timbre model, the control feature corresponding to the frame position within the phoneme is computed with a nonlinear mapping depending on the length of the phoneme. The edges of a phoneme (where transitions are more likely) will maintain their original rate, while the more stable center parts will be expanded.

4 Complexity Analysis

Training the existing NPSS takes around 10 hours on a single Titan X Pascal GPU with approximately 30 minutes of training data. We anticipate similar computational times for our baseline model. In order to expand the existing system to perform transfer learning, the system has to be trained with speaker-specific information. This will result in the dimension of input features to be increased by at least a factor of two. Increasing the feature dimensionality will result in an increase in training and generation times, and may be a key computational bottleneck in our system.

5 Major Challenges

5.1 Data Collection

In general, one of the major challenges in singing processing tasks is the limited amount of data available for training and testing. Existing open datasets are varied in terms of singers and style and often do not fully cover the wide range of traits of the human singing voice. In particular, singing voice synthesis requires a dataset of high quality acapella recordings of a single singer along with the corresponding musical sore and lyrics annotated at the phoneme level. Since very few of these datasets are publicly available, we will need to manually construct a dataset in one of the following ways:

1. Use an existing dataset of acapella recordings, such as MUS 2018, MSD100, and DSD100, and automatically obtain time-aligned, phoneme-level lyric transcriptions using Gentle, or

2. Collect "stripped down" or soloed vocal tracks of singers from YouTube, generate the music scores via Sibelius AudioScore or AthemScore, and phonetically transcribe the lyrics using Gentle.

Although the above data collection options may prove tedious, we only need to train our system on approximately 30 minutes of data to synthesize a natural-sounding singing voice. As explained in the model overview and algorithm sections above, the existing NPSS pitch and timbre models significantly reduce the amount of required training data compared to other methods. In addition, we can benchmark our model's performance on our dataset by also training it on the NIT-SONG070-F001 dataset (`http://hts.sp.nitech.ac.jp/archives/2.3/HTS-demo_NIT-SONG070-F001.tar.bz2`), one of the datasets used for the existing NPSS. The publicly available dataset contains 31 studio quality recordings of Japanese nursery rhymes along with phoneme and note level annotations.

5.2 Speaker Representation

Since our proposed system involves transfer learning to synthesize singing voices for unseen singers/speakers, we need to devise a method of representing singers/speakers during the training phase. There are several methods that we can employ and test.

One option is to use a one-hot vector for singer/speaker encoding. This is the most intuitive or "naive" approach. Another approach is to use an i-vector to represent a singer/speaker, which is a widely used baseline model in modern speaker identification systems. A third method is to employ an autoencoder to learn singer/speaker embeddings at the bottleneck layer.

An advantage to using an autoencoder is that we can integrate this model with our current NPSS system so that the synthesizer and singer/speaker representer can be trained end-to-end. We expect this change to improve speaker-specific generation. More interestingly, we also expect the modification to change the singer/speaker representation itself - since many low level audio features are handled by the WaveNet architectures in our pitch and timbre models, the autoencoder should be able to omit these and learn more high-level abstract information.

5.3 Overfitting: Exposure bias and ignorance on control inputs

While the NPSS generation process is autoregressive, during training ground-truth past samples are used rather than using past predictions. One result-

ing issue is exposure bias, which means that the model becomes biased to the ground-truth samples that it is exposed to during training, causing accumulative errors in autoregressive generation steps. Another source of overfitting is that the model's predictions will only rely on past timestamps and ignore the control inputs. Both issues will cause an unnatural-sounding singing voice over time.

One way to address this overfitting problem is to increase the size of the dataset. Another option is to regularize the network by using a denoising objective:

$$L = -log\, p(\mathbf{x}_t \,|\, \tilde{\mathbf{x}}_{<t}, \mathbf{c}) \tag{4}$$

with

$$\tilde{\mathbf{x}}_{<t} \sim p(\tilde{\mathbf{x}}_{<t} \,|\, \mathbf{x}_{<t}) \tag{5}$$

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

$$p(\tilde{\mathbf{x}} \,|\, \ \mathbf{x}) = N(\tilde{\mathbf{x}}; \mathbf{x}, \lambda I) \tag{6}$$

and λ is noise level. That is, Gaussian noise is added to the input of the network while the network is trained to predict the uncorrupted target. Therefore, we need to apply post-processing to the predicted output distribution to reduce the generated noisy output.

6 Modeling Tasks

Our proposed system models the human singing voice via three models: a phonetic timing model, a pitch model, and a timbre model. The details of each model are described in previous sections.

7 Human Factors

We plan to have users evaluate our system by conducting a MUSHRA style listening test and collecting mean opinion scores (MOS). The listening test will have participants rate our model's synthesized singing versus natural singing in addition to how well our model converts the speaking voice of a new speaker to a singing voice.

8 Rough schedule

1. Collect and clean dataset (First 1-2 weeks)
2. Linguistic feature extraction (First 1-2 weeks)

3. Construct the baseline model (Within 3 weeks)

Task	Members
Implement general WaveNet architecture	Mu, Shiyu
Implement timbre model and algorithms	James
Implement pitch model and algorithms	Sharada
Implement phonetic timing model and algorithms	Yixin

4. Train and test the baseline model (Within 4 weeks)

Task	Members
Train and test timbre model and algorithms on constructed dataset	James
Train and test pitch model and algorithms on constructed dataset	Sharada
Train and test phonetic timing model and algorithms on constructed dataset	Yixin
Train and test all models on the NIT-SONG070-F001 dataset	Mu, Shiyu

5. Transfer learning for new singers (Within 8 weeks)

Task	Members
Begin preparing demo	James
Train and test models with autoencoder implementation	Mu, Sharada
Train and test models with one-hot implementation	Shiyu, Yixin

9 Final Test Set-Up

We will evaluate the performance of our NPSS system using the following quantitative metrics: Mel-Cepstral Distortion (MCD), Band Aperiodicity Distortion (BAPD), Modulation Spectrum (MS) for Mel-Generalized Coefficients (MGC), Voiced/unvoiced decision metrics , Timing metrics, F0 metrics, Modulation Spectrum (MS) for log F0. We will compare our model's performance to the results of the existing NPSS system by training and testing on the NIT-SONG070-F001 dataset. We plan to have users evaluate our system by conducting a MUSHRA style listening test and collecting mean opinion scores (MOS), as described in the human factors section above.

10 Existing codebase

We have an existing codebase for the WORLD parametric vocoder as well as a basic WaveNet architecture implementation (specifically applied to speech-related tasks). However, we will need to implement the phonetic timing, pitch, and timbre models of the existing NPSS system ourselves in addition to the transfer learning task.

References

[1] M. Blaauw and J. Bonada, "A neural parametric singing synthesizer modeling timbre and expression from natural songs," *Applied Sciences*, vol. 7, Dec 2017.

[2] 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," 2016.

[3] 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. E99.D, no. 7, p. 1877–1884, 2016.

[4] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. L. Moreno, and Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," 2018.

[5] S. O. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou, "Neural voice cloning with a few samples," 2018.

[6] E. Nachmani, A. Polyak, Y. Taigman, and L. Wolf, "Fitting new speakers based on a short untranscribed sample," 2018.

[7] R. M. Ochshorn and M. Hawkins, "Gentle forced aligner (computer program)," 2017.