DDSP-BASED SINGING VOCODERS: A NEW SUBTRACTIVE-BASED SYNTHESIZER AND A COMPREHENSIVE EVALUATION

Da-Yi Wu1∗ Wen-Yi Hsiao2∗ Fu-Rong Yang3∗ Oscar Friedman4
Warren Jackson5 Scott Bruzenak4 Yi-Wen Liu3 Yi-Hsuan Yang1,2

1 Academia Sinica, 2 Taiwan AI Labs, 3 National Tsing Hua Univ., 4 470 Music Group, 5 PARC
{ericwudayi2, s101062219, fjbcrs34}@gmail.com

ABSTRACT

A vocoder is a conditional audio generation model that converts acoustic features such as mel-spectrograms into waveforms. Taking inspiration from Differentiable Digital Signal Processing (DDSP), we propose a new vocoder named SawSing for singing voices. SawSing synthesizes the harmonic part of singing voices by filtering a sawtooth source signal with a linear time-variant finite impulse response filter whose coefficients are estimated from the input mel-spectrogram by a neural network. As this approach enforces phase continuity, SawSing can generate singing voices without the phase-discontinuity glitch of many existing vocoders. Moreover, the source-filter assumption provides an inductive bias that allows SawSing to be trained on a small amount of data. Our evaluation shows that SawSing converges much faster and outperforms state-of-the-art generative adversarial network- and diffusion-based vocoders in a resource-limited scenario with only 3 training recordings and a 3-hour training time.

1. INTRODUCTION

Singing voice synthesis (SVS) aims to generate human-like singing voices from musical scores with lyrics [1–9]. State-of-the-art (SOTA) voice synthesis techniques involve two stages: acoustic feature modeling from musical scores and audio sample reconstruction via a so-called “vocoder.” A neural vocoder takes an acoustic feature such as mel-spectrogram as input and outputs a waveform using deep learning networks [10–22]. However, phase discontinuities within partials often occur due to the difficulty of reconstructing realistic phase information from a mel-spectrogram. This may lead to a short-duration broadband transient perceived as “glitch” or “voice tremor,” which is more audible during long utterances commonly found in singing [18], as exemplified in Figure 1.

*Equal contribution. Preliminary work was done while Wu was a remote intern working with Friedman at 470 Music Group, LLC.


Figure 1: The magnitude spectrograms of a long utterance of an original recording (‘ground truth’) and those reconstructed by two widely-used neural vocoders, Parallel WaveGAN (PWG) [14] and HiFi-GAN [15], and the proposed SawSing. Each vocoder is trained on 3 hours of recordings from a female singer until convergence. We see glitches in the results of PWG and HiFi-GAN.

Differentiable Digital Signal Processing (DDSP) [23] introduces a new paradigm for neural audio synthesis. It incorporates classical digital signal processing (DSP) synthesizers and effects as differentiable functions within a neural network (NN), and combines the expressiveness of an NN with the interpretability of classical DSP. The use of phase-continuous oscillators is a potential solution to the phase problem from which regular neural vocoders suffer, and the strong inductive bias of this approach may obviate the need of large training data. Furthermore, DDSP has already succeeded in achieving sound synthesis of, and timbre transfer between, monophonic instruments [23–30]. These motivate us to explore whether the DDSP approach can be applied to build a singing vocoder.

This paper proposes SawSing, a DDSP-based singing vocoder which reconstructs a monophonic singing voice from a mel-spectrogram. The architecture of SawSing similarly consists of an NN and classical DSP components; unlike DDSP, its DSP portion is a subtractive harmonic synthesizer which filters a sawtooth waveform containing all possible harmonic partials, plus a subtractive noise synthesizer which filters uniform noise. The sawtooth signal enforces phase continuity within partials, thereby avoiding the glitches. Moreover, the partials of a sawtooth signal are guaranteed to be in phase, so it also enforces the phase coherence between partials, intrinsic to human voices. The function of the NN, on the other hand, is to infer from the mel-spectrogram the fundamental frequency (f0) of the sawtooth signal and the filter coefficients of the harmonic and noise synthesizers for each time frame.
In our experiments, we use data from two singers (each three hours) of the MPop600 Mandarin singing corpus [31]. We compare the performance of SawSing with the neural source-filter (NSF) model [12], two existing DDSP-based synthesizers, the original additive-based DDSP [23] and the differentiable wavetable synthesizer [27], and a few famous neural vocoders, i.e., two generative adversarial network (GAN)-based models [5, 14] and a diffusion-based model [21]. We consider both a regular scenario where the vocoders are trained for days using the 3-hour dataset, and a resource-limited scenario with constraints on training data and training time. Our experiments show that SawSing converges much faster and outperforms the other vocoders in the resource-limited scenario.

The main contribution of the paper is two-fold. First, we show that despite differences between instrumental sounds and singing voices [32], the classic idea of subtractive synthesis [33, 34] can be applied to singing voices using the DDSP approach. Second, we provide empirical evidences showing that DDSP-based vocoders can compare favorably with sophisticated, SOTA neural vocoders. Furthermore, since DDSP-based vocoders are lightweight and training-efficient, they have the potential to be used in creative and real-time scenarios of singing expression with limited training data of a target singing voice [36, 37].

We open source our code at https://github.com/YatingMusic/ddsp-singing-vocoders/. For audio examples, visit our demo webpage https://ddspvocoder.github.io/ismir-demo/.

2. BACKGROUND

Neural vocoders often aim to reconstruct a waveform \( y \in \mathbb{R}^{1 \times T} \) from an input mel-spectrogram \( X \in \mathbb{R}^{M \times N} \):

\[
y = f_{\text{vocoder}}(X),
\]

where \( M, N, T \) denote respectively the number of mel filter banks, spectral frames, and time-domain samples. The conversion from \( X \) to \( y \) can be done, for example, by up-sampling \( X \) multiple times through transposed convolutions until the length of the output sequence matches the temporal resolution of the raw waveform [13, 15]. As usual reconstruction loss functions such as mean-square errors cannot reflect the perceptual quality of the reconstruction, GAN-based approaches [13–15] learn discriminators to better guide the learning process of the generator (i.e., \( f_{\text{vocoder}} \)). Newer diffusion-based approaches [20, 21] avoid the use of discriminators and learn to convert white Gaussian noises \( z \in \mathbb{R}^{1 \times T} \) (i.e., of the same length as \( y \)) into structured waveform \( y \) through a denoising-like Markov chain, using \( X \) as a condition. The mapping process between \( X \) and \( y \) of such neural vocoders appears to be a black box that is hard to interpret. However, given sufficient training data (e.g., recordings amounting to 24 hours [15, 20, 21, 38] or 80 hours [18]) and training time (e.g., days), SOTA neural vocoders can reconstruct the waveforms with high fidelity.

The majority of neural vocoders, however, have been originally developed for speech. When the rate of utterances is fast, as is common in speech, the glitches resulting from the phase discontinuities within partials may be perceptually masked by the natural transients of the voice. However, during singing, where long utterances are common, these discontinuities are more audible.

To improve the performance of GAN-based vocoders for singing voices, the idea of incorporating the f0 information has been explored recently. PeriodNet [16] uses f0 to create sine excitation as input to Parallel WaveGAN (PWG) [14] to model the periodic part of human voices. Guo et al. [17] further filter such an f0-driven excitation signal with a linear time-variant finite impulse response (LTV-FIR) filter whose coefficients are estimated from the input mel-spectrogram, and use the resulting “harmonic signal” as input to PWG and MelGAN [13]. SingGAN [18] uses more complicated “adaptive feature learning” layers to incorporate the f0. These models were shown to outperform older GAN-based vocoders such as PWG and MelGAN in listening tests, but no evaluations against the newest GAN-based vocoder HiFi-GAN [15] and diffusion-based vocoders were reported. Moreover, their evaluation did not consider resource-limited scenarios.

We propose in this paper a radically different approach that uses traditional DSP synthesizers (instead of upsampling convolutions) as the backbone for \( f_{\text{vocoder}} \). While the ideas in DDSP have flourished and been applied to synthesizing not only instrumental sounds [23–27], but also audio effects [39–43], their application to singing synthesis remains under-explored. The only exception, to our knowledge, is the preliminary work presented by Alonso and Erkut [44], which employed exactly the same additive synthesizer as the original DDSP paper [23]. However, they did not compare the performance of their vocoder with any other vocoders. Our work extends theirs by using a subtractive harmonic synthesizer instead, with comprehensive performance evaluations against SOTA neural vocoders such as HiFi-GAN and FastDiff [21].

We note that, while a DDSP-based vocoder may solve the glitch problems by inducing continuous phase hypothesis using a harmonic synthesizer, this hypothesis may constrain the model learning ability. Experiments reported in this paper are needed to study its performance.

Publicly-available training corpora for singing tend to be much smaller than those for speech [31, 45] (often ≤10 hours). Therefore, besides tackling the glitch problem, our premise is that SawSing can learn faster than prevalent neural vocoders without a large training corpus, due to its strong inductive bias. Moreover, the success of SawSing may pave the way for the exploration of other advanced DSP components for singing synthesis in the future.

3. ORIGINAL DDSP-ADD SYNTHESIZER

The idea of DDSP is to use DSP synthesizers to synthesize the target audio, with the parameters of the synthesizers \( \Phi \).
inferred from the mel-spectrogram with an NN. Namely,

\[ y = f_{DSP}(\Phi), \quad \Phi = f_{NN}(X). \]  

(2)

The original DDSP model [23], referred to as DDSP-Add below, adopts the harmonic-plus-noise model for synthesis [46] and decomposes a monophonic sound into a periodic (harmonic) component \( y_n \) and a stochastic (noise) component \( y_{\text{ns}} \), i.e., \( y = y_n + y_{\text{ns}} \), and reconstructs them separately with an additive harmonic oscillator (thus the name “Add”) and a subtractive noise synthesizer. 2 The former computes \( y_n \) as a weighted sum of \( K \) sinusoids corresponding to the \( f0 \) and its integer multiples up to the Nyquist frequency (for anti-aliasing), for \( t \in [1, T] \):

\[ y_n^{\text{DDSP-Add}}(t) = A(t) \sum_{k=1}^{K} c_k(t) \sin(\phi_k(t)), \]  

(3)

where \( A(t) \) is the global amplitude corresponding to the time step \( t \), \( c_k(t) \) is the amplitude of the \( k \)-th harmonic satisfying \( \sum_{k=1}^{K} c_k(t) = 1 \), \( c_k(t) \geq 0 \), and the instantaneous phase \( \phi_k(t) \) is computed by integrating the instantaneous frequency \( k f_0(t) \), i.e., \( \phi_k(t) = 2\pi \sum_{\tau=0}^{t} k f_0(\tau) + \phi_{0,k} \), with \( \phi_{0,k} \) initial phase, set to zero. The parameters \( A, c_k, f_0 \) are estimated by \( f_{NN} \) for each frame \( i \in [1, N] \) and then upsampled to the time-domain with linear interpolation. On the other hand, \( y_{\text{ns}} \) is obtained by convolving a uniform noise signal \( \zeta \) ranging from \(-1 \) to \(1 \) (with the same length as a frame) with an LTV-FIR filter \( \psi_i(i) = 1_{\text{LTV-FIR filter}} \) estimated per frame:

\[ \hat{y}_{\text{ns}}^{\text{SawSing}}(i) = \zeta \ast \psi_i(i). \]  

(4)

The final \( y_{\text{ns}} \) is obtained by overlap-adding sequence of segments \( y_{\text{ns}}(i) \) for the frames \( i = 1 \ldots N \). Jointly, the parameters \( \Phi := \{ A(i), \{ c_k(i) \}_{k=1}^{K}, f_0(i), \psi_i(i) \}_{i=1}^{N} \) are estimated from the mel-spectrogram \( X \) per frame by \( f_{NN} \), which is a small network with few parameters.

Engel et al. [23] showed that DDSP-Add can synthesize realistic violin sounds with only 13 minutes of expressional sources such as oscillators or wavetables, while a subtractive noise synthesizer creates sounds by using filters to shape a source signal, typically with rich harmonics, such as a square or sawtooth wave [46].

Second, \( \hat{y}_{\text{ns}} \) is treated as the “excitation signal” and shaped into the desirable \( y_n \) by means of an LTV-FIR filter \( \psi_i(i) \in \mathbb{R}^{L_0} \) (that is different from \( \psi_i(i) \)). To apply the filter, we extract the segment of \( \hat{y}_{\text{ns}} \) corresponding to the same frame \( i \) and multiply its short-time Fourier Transform (STFT) element-wise with the STFT of \( \psi_i(i) \) in the frequency domain, before converting it back to the time domain with the inverse STFT and overlap-adding. SawSing uses the same subtractive noise synthesizer as DDSP-Add. Therefore, the parameters to be estimated from \( X \) by \( f_{NN} \) are \( \Phi^{\text{SawSing}} := \{ f_0(i), \psi_i(i), \psi_i(i) \}_{i=1}^{N} \).

We observe that to compute \( y_n \), DDSP-Add learns NN to attenuate each of the \( k \) source harmonics individually (i.e., with \( c_k \)), while SawSing entails a source-filter model [12], using the \( f0 \)-constrained sawtooth signal in Eqn. (5) as the excitation source and a time-varying filter \( \psi_i(i) \) decided by the NN for spectral filtering. The filter coefficients correspond to formants produced by the vocal folds and do not correlate with \( f0 \).

Besides differences in the harmonic synthesizer, SawSing also uses a different loss function from DDSP-Add. For monophonic instrumental sounds, Engel et al. [23] showed it effective to use the multi-resolution STFT (MSSTFT) loss as the reconstruction loss for training. This loss considers the difference between the magnitude spectrograms of the target and synthesized audio, denoted as \( S_j \) and \( \hat{S_j} \), below for \( J \) different resolutions.

\[ l_{\text{MSSTFT}} = \sum_{j=1}^{J} \| S_j - \hat{S}_j \|_1 + \| \log(S_j) - \log(\hat{S}_j) \|_1. \]  

(6)

For singing voices, however, we found that MSSTFT loss alone cannot train adequately. We introduce an additional \( f0 \)-related loss term to facilitate learning:

\[ l_{f0} = \| \log(f_0) - \log(\hat{f}_0) \|_1, \]  

(7)

where the target \( f0 (f_0) \) and the estimated one \( \hat{f}_0 \) are both extracted by the WORLD vocoder [47]. Thus, our SawSing loss function becomes \( l_{\text{total}} = l_{\text{MSSTFT}} + l_{f0} \). Moreover, we found that training is unstable unless the gradients between \( f_{DSP} \) and the head of \( f_{NN} \) for \( f0 \) prediction are detatched.

4.1 Implementation Details

First, we resampled the audio recordings to 24 kHz and quantized them to 16 bits. Next we cropped the recordings into 2-second excerpts (i.e., \( T = 48k \)) and extracted 80-band mel-spectrograms from each (\( M = 80 \)), with a Hann window of 1024 samples for STFT and a hop size of 240 samples (i.e., 10ms). Accordingly, we set \( N = 200 \).

We used filter length \( L_n = 256 \) for the harmonic synthesizer for SawSing, and filter length \( L_n = 80 \) for the subtractive noise synthesizers. We used at most \( K = 150 \) sinusoids for SawSing. To avoid sound clipping, we applied a global scaling factor of 0.4 to the sawtooth signal in Eqn. (5) to ensure that the range of the summed sinusoids always lies in \([-1, 1]\).

We chose a lite version of the Conformer architecture for \( f_{NN} \) [48], for its well-demonstrated effectiveness in
capturing both local and global information in a sequence of acoustic features in speech tasks. It consists of a pre-net (shallow 1D convolution with ReLU activation and group normalization), a self-attention stack (3 layers), a convolution stack (2 layers) with post layer normalization, and a final linear layer whose output dimension is equal to the number of synthesis coefficients. We used the Adam optimizer with 0.002 learning rate.

While the original DDSP-Add paper [23] uses \( J = 6 \) for MSSTFT, we found setting \( J = 4 \) to be sufficient in our implementation. Specifically, we used four different FFT sizes \((128, 256, 512, 1024)\) with 75% overlapping among adjacent frames. While it is possible to introduce a scaling factor to control the balance between \( l_{\text{MSSTFT}} \) and \( l_{f_0} \), we found doing so does not markedly improve the result.

5. EXPERIMENTAL SETUP

5.1 Baselines

Our evaluation considers in total six baselines. The first three explicitly employ \( f_0 \), while the last three do not.

First, we adopted two existing DDSP-based vocoders, the original additive-based DDSP (DDSP-Add) [23, 44] and the differentiable wavetable synthesizer (DWTS) [27].

**DDSP-Add** replaces the fixed sinusoids in the additive harmonic synthesizer of DDSP-Add by \( K' \) learnable (rather than pre-defined) one-cycle waveforms ("the wavetables") \( w_k \in \mathbb{R}^D, k \in \{1, K'\} \), to gain flexibility to model a wider variety of sounds (but only tested on instrumental sounds in [27]). Mathematically, \( y_{\text{DDSP-Add}}^D(t) = A(t) \sum_{k=1}^{K'} c_k(t) \sigma(w_k, \phi_\sigma(t)) \), where \( \sigma \) is an indexing function that returns a sample of \( w_k \) according to the instantaneous modulo phase \( \phi_\sigma(t) \) computed from \( f_0(t) \). For fair comparison, we used the same Conformer-like architecture for the \( f_\text{NN} \) for DDSP-Add, DWTS, and SawSing, and the same noise synthesizer. Moreover, while the original DDSP-Add used only \( l_{\text{MSSTFT}} \), thus we used \( l_{\text{total}} = l_{\text{MSSTFT}} + l_{f_0} \) for all three in our implementation. Like SawSing, we set \( K = 150 \) for DDSP-Add. DWTS only needs small \( K' \) as the wavetables are learnable; we set \( K' = 20 \), with wavetable length being \( B = 512 \).

We also employed the neural source-filter (NSF) waveform model [12], which was proposed before the notion "DDSP" was coined [23]. Unlike SawSing, NSF uses unweighted sinusoids (i.e., \( \sum_{k=1}^{K} \sin(\phi_k(t)) \)) as the source signal, and uses stacked dilated-convolution blocks instead of a simple LTV-FIR filter. We adapted the open-source code from the original authors to implement NSF, as well as the following three baselines.

For GAN-based neural vocoders, we used PWG [14] and HiFi-GAN [15], both of which were developed for speech, not singing.\(^3\) PWG is a non-autoregressive version of WaveNet [10] that learns to transform a random noise into target audio with 30 layers of dilated residual convolution blocks, conditioning on the mel-spectrogram.

\(^3\)While we did not consider SingGAN [18] in the evaluation for lack of open source code, we should have included PeriodNet [16] for it seems easy to implement. Unfortunately we are aware of PeriodNet too late.

For HiFi-GAN, we used the most powerful “V1” configuration [15], which converts a mel-spectrogram into a waveform directly via 12 residual blocks. It uses a sophisticated multi-receptive field fusion module in the generator, and multiple multi-scale and multi-period discriminators [15].

An increasing number of diffusion-based vocoders have been proposed in the past two years for speech [19–22]. We adopt as a baseline the FastDiff model [21], which has been shown to beat HiFi-GAN V1 [15] and diffusion-based models WaveGrad [19] and DiffWave [20] in the mean-opinion-score (MOS) of vocoded speech in listening tests. However, while a noise schedule predictor has been devised to reduce the sampling steps of the denoising Markov chain, the inference time of FastDiff is still around 10 times slower than HiFi-GAN, according to [21].

None of these baselines have been trained on MPop600, which is a relatively new dataset. Therefore, we trained all these models from scratch with the MPop600 data.

5.2 Dataset & Scenarios

Our data is from MPop600 [31], a set of accompaniment-free Mandarin singing recordings with manual annotation of word-level audio-lyrics alignment. Each recording covers the first verse and first chorus of a song. We used the data from a female singer (named \( f_1 \)) and a male singer (\( m_1 \)); each has 150 recordings. For each singer, we reserved 3 recordings (totaling 3.4–3.6 minutes in length) as the test set for subjective evaluation, 24 or 21 recordings (around 27–28 minutes) as the validation set for objective evaluation, and used the rest (around 3 hours) as the training set. We trained vocoders for \( m_1 \) and \( f_1 \) independently.

To study the training efficiency of different approaches, we considered the following two scenarios. We used the same validation and test sets for both scenarios.

(a) **Regular** \([3\text{h data, well-trained}]\): we used the full training data to train the vocoders for each singer for up to 2.5 days (i.e., when the training loss of most vocoders converged), and picked the epoch that reaches the lowest validation loss for each vocoder independently. We note that the amount of training time in this "regular" scenario is smaller than those seen in speech vocoders [38], posing challenges for all the considered models.

(b) **Resource-limited** \([3\text{min data, 3h training}]\): in a rather extreme case, we randomly picked 3 recordings from the training set (that collectively cover every phoneme at least once) per singer (3.2–3.4 minutes) for training, using always the epoch at 3-hour training time.

For fair comparison, we train the vocoders of different approaches using a dedicated NVIDIA GeForce RTX 3090 GPU each, fixing the batch size to 16 excerpts.

6. OBJECTIVE EVALUATION

For objective evaluation, we reported the MSSTFT and the mean absolute error (MAE) in \( f_0 \), as well as the Fréchet audio distance (FAD) [49] between the validation data and the reconstructed ones by the vocoders. FAD measures the
Table 1: Objective evaluation results of three existing neural vocoders (the first three), three existing DDSP-based vocoders (middle) and the proposed SawSing vocoder, trained on either a female or a male singer, in either (a) regular scenario [3h data, well-trained] or (b) resource-limited scenario [3min data, 3h training]. RTF stands for real-time factor (the inference time in seconds for a one-second excerpt). In each column, we highlight the best result in bold, the second best underlined.

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>RTF</th>
<th>MSSTFT ↓</th>
<th>MAE-f0 (cent) ↓</th>
<th>FAD ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Female (a)</td>
<td>Male (b)</td>
<td>Female (a)</td>
</tr>
<tr>
<td>FastDiff [21]</td>
<td>15.3M</td>
<td>0.017</td>
<td>14.5</td>
<td>17.9</td>
<td>11.1</td>
</tr>
<tr>
<td>HiFi-GAN [15]</td>
<td>13.9M</td>
<td>0.004</td>
<td>7.13</td>
<td>16.7</td>
<td>7.82</td>
</tr>
<tr>
<td>PWG [14]</td>
<td>1.5M</td>
<td>0.007</td>
<td>7.39</td>
<td>13.0</td>
<td>7.83</td>
</tr>
<tr>
<td>NSF [12]</td>
<td>1.2M</td>
<td>0.006</td>
<td>7.51</td>
<td>10.9</td>
<td>10.2</td>
</tr>
<tr>
<td>DDSP-Add [44]</td>
<td>0.5M</td>
<td>0.003</td>
<td>7.61</td>
<td>9.29</td>
<td>8.37</td>
</tr>
<tr>
<td>DWTS [27]</td>
<td>0.5M</td>
<td>0.019</td>
<td>7.72</td>
<td>9.75</td>
<td>8.83</td>
</tr>
<tr>
<td>SawSing</td>
<td>0.5M</td>
<td>0.003</td>
<td><strong>6.93</strong></td>
<td><strong>8.79</strong></td>
<td><strong>7.76</strong></td>
</tr>
</tbody>
</table>

Table 1: Objective evaluation results of three existing neural vocoders (the first three), three existing DDSP-based vocoders (middle) and the proposed SawSing vocoder, trained on either a female or a male singer, in either (a) regular scenario [3h data, well-trained] or (b) resource-limited scenario [3min data, 3h training]. RTF stands for real-time factor (the inference time in seconds for a one-second excerpt). In each column, we highlight the best result in bold, the second best underlined.

Table 1 also shows that the performance gap between scenarios (a) and (b) in all the three metrics tend to be greater for the diffusion- and GAN-based vocoders than for NSF and the DDSP-based vocoders. Besides, among the evaluated models, the performance gap between (a) and (b) is the smallest in the result of SawSing. In the resource-limited scenario (b), the FAD of HiFi-GAN reaches only 3.50 and 10.5 for the female and male singers, respectively, while the FAD of SawSing can be lower than 1.0. This demonstrates that a strong inductive bias like those employed in NSF and the DDSP-based vocoders is helpful in scenarios with limited training data and training time.

Table 1 also displays the real-time factor (RTF) of the models when being tested on a single NVIDIA 3090 GPU. We see that SawSing and DDSP-Add have the lowest RTF (i.e., run the fastest), followed by HiFi-GAN.

According to Table 1, HiFi-GAN performs the best on average among the first three vocoders across scenarios and singers. NSF and the DDSP-based vocoders obtain comparable scores, but DWTS is notably slower. Hence, we pick HiFi-GAN, NSF, DDSP-Add and SawSing to be further evaluated in the user study below.

7. SUBJECTIVE EVALUATION

We conducted an online study to evaluate the performance of the 4 selected models. We had 2 sets of questionnaires, one for the female and the other for the male singer. For each singer, we prepared 8 clips from the 3 testing recordings (i.e., totally unseen at training/validation time), each clip corresponding to the singing of a full sentence. We

---

4In the original paper [21], FastDiff was trained on 24 hours of speech data from a female speaker [38], using 4 NVIDIA V100 GPUs. We tried DiffWave [20] but it converged similarly slow on our data.
let the vocoders trained in scenario (a) to reconstruct the waveforms from the mel-spectrograms of 4 of the clips, and those of (b) for the other 4 clips. A human subject was requested to use a headset to listen to 5 versions of each of the clip, namely the original ‘ground truth’ recording and the reconstructed ones by the 4 selected models, with the ordering of the 5 versions randomized, the ordering of the 8 clips randomized, and not knowing the scenario being considered per clip. The loudness of the audio files were all normalized to −12dB LUFS beforehand using pyloudnorm [51]. After listening, the subject gave an opinion score from 1 (poor) to 5 (good) in a 5-point Likert scale to rate the audio quality for each audio file.

Figure 3 shows the MOS from 23 anonymized participants for the female and 18 participants for the male singer. In scenario (a), we see that the MOS of the vocoders, including the SOTA HiFi-GAN, mostly reaches 2–3 only, suggesting that training a vocoder on 3-hour data is already challenging. As HiFi-GAN involves a complicated GAN training and much more parameters, its MOS turns out to be significantly lower than those of NSF and the DDSP-based vocoders (p-value<0.05 in paired t-test). Interestingly, while there is no statistical difference among the MOS of NSF, DDSP-Add and SawSing for the male singer in scenario (a), DDSP-Add unexpectedly outperforms both NSF and SawSing by a large margin, with statistically significant difference (p-value<0.05).

Listening to the result of SawSing reveals that its output contains an audible electronic noise, or “buzzing” artifact, notably when singers emphasize the airflow with breathy sounds and for unvoiced consonants such as /s/ and /t/. DDSP-Add is free of such an artifact. As shown in Figure 4, such artifact appears to due to redundant harmonics generated by SawSing that “connect” the harmonics of two adjacent phonemes at its harmonic signal $x_n$, for breathing and unvoiced consonants. This may be due to the limited capacity of the LTI-FIR filter of SawSing in distinguishing between the nuances of voiced (V) and unvoiced (NV) components during sound transients, modeling a transient even as a harmonic signal. Unfortunately, it seems that this artifact cannot be reflected in any training loss functions (and objective metrics) we considered, so the network fails to take it into account while updating the parameters. Furthermore, human ears are sensitive to such an artifact, contributing to the lower MOS of SawSing compared to DDSP-Add, despite that SawSing might perform better in other phonemes and long utterances.

Figure 3 also shows that the DDSP-based vocoders do outperform HiFi-GAN greatly in the resource-limited scenario (b) with only 3 training recordings, nicely validating the training efficiency of the DDSP-based vocoders. While the MOS of either DDSP-Add or SawSing is above 2; that of HiFi-GAN is only around 1, i.e., its generation is barely audible. Moreover, SawSing outperforms DDSP-Add in this scenario for both singers, with significant MOS difference for the male singer (p-value<0.05), though not for the female singer. This shows that, despite the buzzing artifact, the training efficiency of SawSing can give it an edge over other vocoders in resource-limited applications.

Inspired by [5], we implement a postprocessing method that uses Parselmouth [52] to get V/NV flags and sets the harmonic synthesizer amplitudes to zero for the NV portions. This removes much of the artifact (see the demo page). We share the code on our GitHub repo. Future work can incorporate the V/NV flags at the training phase.

8. CONCLUSION

In this paper, we have presented SawSing, a new DDSP-based vocoder that synthesizes an audio via the summation of a harmonic component obtained from filtered sawtooth waves and a stochastic component modeled by filtered noise. Moreover, we presented objective and subjective evaluations complementing the lack of experiments in the recent work of Alonso and Erkut [44], demonstrating for the first time that both SawSing and DDSP-Add [23, 44] compare favorably with SOTA general-purpose neural vocoders such as HiFi-GAN [15] and FastDiff [21] for singing voices in a regular-resource scenario, and has a great performance margin in a resource-limited scenario.

Future work can improve the harmonic filter of SawSing to resolve the artifact, and use lighter-weight non-causal convolutions [53] for $f_{NN}$ for real-time applications. We can also implement SawSing as a VST audio plugin for usages in creative workflows and music production [54].
9. ACKNOWLEDGEMENT

We are grateful to Rongjie Huang and Yi Ren for sharing with us the code of FastDiff, and Shuhei Imai for helping proofread the paper. We also thank the anonymous reviewers for their constructive feedbacks. Our research is funded by grants NSTC 109-2628-E-001-002-MY2 and NSTC 109-2221-E-007-094-MY3 from the National Science and Technology Council of Taiwan.

10. REFERENCES


