inferred from the mel-spectrogram with an NN. Namely,

$$\mathbf{y} = f_{\text{DSP}}(\Phi)\,, \quad \Phi = f_{\text{NN}}(\mathbf{X})\,. \tag{2}$$

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 $\mathbf{y}_\text{h}$ and a stochastic (noise) component $\mathbf{y}_\text{n}$, i.e., $\mathbf{y} = \mathbf{y}_\text{h} + \mathbf{y}_\text{n}$, and reconstructs them separately with an *additive* harmonic oscillator (thus the name "-Add") and a *subtractive* noise synthesizer. [2] The former computes $\mathbf{y}_\text{h}$ as a weighted sum of K sinusoids corresponding to the f0 and its integer multiples up to the Nyquist frequency (for anti-aliasing), for $t \in [1, T]$:

$$\mathbf{y}_\text{h}^{\text{DDSP-Add}}(t) = A(t) \sum_{k=1}^{K} c_k(t)\, \sin(\phi_k(t))\,, \tag{3}$$

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 $kf_0(t)$, i.e., $\phi_k(t) = 2\pi \sum_{\tau=0}^{t} kf_0(\tau) + \phi_{0,k}$, 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, $\mathbf{y}_\text{n}$ is obtained by convolving a uniform noise signal ζ ranging from -1 to 1 (with the same length as a frame) with an LTV-FIR filter $\psi_\text{n}(i) \in \mathbb{R}^{L_\text{n}}$ estimated per frame:

$$\bar{\mathbf{y}}_\text{n}(i) = \zeta * \psi_\text{n}(i)\,. \tag{4}$$

The final $\mathbf{y}_\text{n}$ is obtained by overlap-adding sequence of segments $\bar{\mathbf{y}}_\text{n}(i)$ for the frames $i = 1 \ldots N$. Jointly, the parameters $\Phi := \{A(i), \{c_k(i)\}_{k=1}^{K}, f_0(i), \psi_\text{n}(i)\}_{i=1}^{N}$ are estimated from the mel-spectrogram $\mathbf{X}$ 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 expressive solo violin performances as training data. Alonso and Erkut [44] employed DDSP-Add for singing synthesis, but with limited performance evaluation.

4. PROPOSED SAWSING VOCODER

Under the same harmonic-plus-noise signal model [46], SawSing modifies the the harmonic synthesizer of DDSP-Add [23] with two ideas. First, given the f0 estimated from $\mathbf{X}$, SawSing approximates $\mathbf{y}_\text{h}$ by a sawtooth signal, which contains an equal number of even and odd harmonics with decaying magnitudes, dropping the coefficients A and c_k:

$$\widetilde{\mathbf{y}_\text{h}}^{\text{SawSing}}(t) = \sum_{k=1}^{K} \frac{1}{k} \sin(\phi_k(t))\,. \tag{5}$$

[2]The terms "additive" and "subtractive" are used to describe how a signal is synthesized. An additive synthesizer generates sounds by combining multiple sources such as oscillators or wavetables, while a subtractive synthesizer creates sounds by using filters to shape a source signal, typically with rich harmonics, such as a square or sawtooth wave [46].

Second, $\widetilde{\mathbf{y}_\text{h}}$ is treated as the "excitation signal" and shaped into the desirable $\mathbf{y}_\text{h}$ by means of an LTV-FIR filter $\psi_\text{h}(i) \in \mathbb{R}^{L_\text{h}}$ (that is different from $\psi_\text{n}(i)$). To apply the filter, we extract the segment of $\widetilde{\mathbf{y}_\text{h}}$ corresponding to the same frame i and multiply its short-time Fourier Transform (STFT) element-wise with the STFT of $\psi_\text{h}(i)$ 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 $\mathbf{X}$ by f_NN are $\Phi^{\text{SawSing}} := \{f_0(i), \psi_\text{h}(i), \psi_\text{n}(i)\}_{i=1}^{N}$.

We observe that to compute $\mathbf{y}_\text{h}$, 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 $\psi_\text{h}(i)$ 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 $\mathbf{S}_j$ and $\widehat{\mathbf{S}_j}$ below, for J different resolutions.

$$l_{\text{MSSTFT}} = \sum_{j=1}^{J} \|\mathbf{S}_j - \widehat{\mathbf{S}_j}\|_1 + \|\log(\mathbf{S}_j) - \log(\widehat{\mathbf{S}_j})\|_1\,. \tag{6}$$

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

$$l_{f_0} = \|\log(f_0) - \log(\widehat{f_0})\|_1\,, \tag{7}$$

where the target f0 (f_0) and the estimated one ($\widehat{f_0}$) are both extracted by the WORLD vocoder [47]. Thus, our Sawsing loss function becomes $l_\text{total} = l_\text{MSSTFT} + l_{f_0}$. Moreover, we found that training is unstable unless the gradients between f_DSP and the head of f_NN for f0 prediction are detached.

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 $L_\text{h} = 256$ for the harmonic synthesizer for SawSing, and filter length $L_\text{n} = 80$ 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_{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 f0, 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]. **DWTS** replaces the fixed sinusoids in the additive harmonic synthesizer of DDSP-Add by K' learnable (rather than pre-defined) one-cycle waveforms ("the wavetables") $\mathbf{w}_k \in \mathbb{R}^B, k \in [1, K']$, to gain flexibility to model a wider variety of sounds (but only tested on instrumental sounds in [27]). Mathematically, $\mathbf{y}_{\text{h}}^{\text{DWTS}}(t) = A(t) \sum_{k=1}^{K'} c_k(t)\sigma(\mathbf{w}_k, \phi_\pi(t))$, where σ is an indexing function that returns a sample of $\mathbf{w}_k$ according to the instantaneous modulo phase $\phi_\pi(t)$ computed from $f_0(t)$. For fair comparison, we used the same Conformer-like architecture for the f_{NN} for DDSP-Add, DWTS, and SawSing, and the same noise synthesizer. Moreover, while the original DDSP-Add used only l_{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 `f1`) and a male singer (`m1`); each has 150 recordings. For each singer, we reserved 3 recordings (totalling 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 `m1` and `f1` 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* [3h 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* [3min 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.

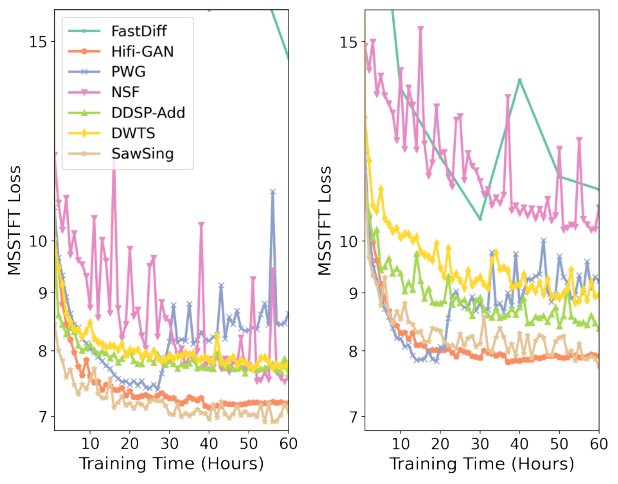
For fair comparison, we train the vocoders of different approaches using a dedicated NVIDIA GeForce RTX 3090 GPU each, fixing the batch size to 16 excerpts.

6. OBJECTIVE EVALUATION

For objective evaluation, we reported the MSSTFT and the mean absolute error (MAE) in f0, as well as the Fréchet audio distance (FAD) [49] between the validation data and the reconstructed ones by the vocoders. FAD measures the

Model	Para-	RTF	MSSTFT ↓				MAE-f0 (cent) ↓				FAD ↓			
	meters		Female		Male		Female		Male		Female		Male	
			(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)
FastDiff [21]	15.3M	0.017	14.5	17.9	11.1	16.9	<u>31</u>	110	48	131	2.29	7.40	3.53	10.0
HiFi-GAN [15]	13.9M	0.004	<u>7.13</u>	16.7	<u>7.82</u>	18.9	34	247	34	433	0.59	3.50	0.51	10.5
PWG [14]	1.5M	0.007	7.39	13.0	7.83	14.8	35	129	<u>29</u>	126	<u>0.36</u>	6.15	2.56	6.29
NSF [12]	1.2M	0.006	7.51	10.9	10.2	13.4	37	**50**	30	<u>82</u>	0.49	3.73	2.08	4.83
DDSP-Add [44]	0.5M	0.003	7.61	<u>9.29</u>	8.37	<u>12.1</u>	**28**	<u>70</u>	**24**	**80**	0.56	<u>0.92</u>	1.06	<u>2.09</u>
DWTS [27]	0.5M	0.019	7.72	9.75	8.83	13.0	**28**	127	**24**	662	0.60	2.98	<u>0.36</u>	8.58
SawSing	0.5M	0.003	**6.93**	**8.79**	**7.76**	**11.7**	32	76	30	**80**	**0.12**	**0.38**	**0.22**	**0.59**

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.



(i) female (ii) male

Figure 2: The MSSTFT loss on the validation set of different vocoders in the 3-hour data & well-trained scenario.

similarity of the real data distribution and generated data distribution in an embedding space computed by a pre-trained VGGish-based audio classifier, and may as such better reflect the perceptual quality of the generated audio.

Figure 2 shows the validation MSSTFT loss as a function of the training time in the regular scenario for the two singers. In Figure 2(i), SawSing converges faster than the other models and reaches the lowest loss (i.e., 6.93), followed by HiFi-GAN and PWG. While DDSP-add and DWTS converge similarly fast as SawSing, they reach at a slightly higher loss (around 7.50). In Figure 2(ii), Sawing, HiFi-GAN and PWG perform comparably in the first 20 hrs. For both singers, PWG overfits when the training time gets too long. Moreover, FastDiff converges the most slowly, followed by NSF. Even with 60-hour training time, the MSSTFT of FastDiff remains to be high (e.g., 14.5 for the female singer), suggesting that our training data might not be big enough for this diffusion-based model.[4]

Table 1 shows the scores in all the three metrics on the validation set for both scenarios, using the epoch (a) at the lowest validation loss or (b) at 3h training. Despite having few trainable parameters, SawSing performs the best in MSSTFT and FAD across both scenarios and both singers, demonstrating its effectiveness as a singing vocoder. For scenario (b), DDSP-Add obtains the second-lowest MSSTFT and FAD across the two singers.

For MAE-f0, SawSing attains scores comparable to the best baseline models. The average MAE-f0 of SawSing is less than a semitone (100 cents). Future work can use a specialized module (e.g., [50]) for the f0 prediction part in f_{NN} of SawSing to further improve the MAE-f0.

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 vocodoers 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

[4]In 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.

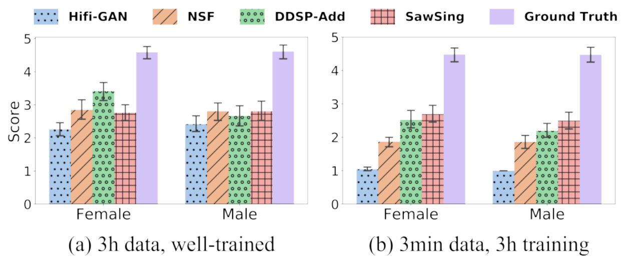


Figure 3: MOS with 95% confidence intervals for subjective evaluation of vocoders trained in the two scenarios.

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_h 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.

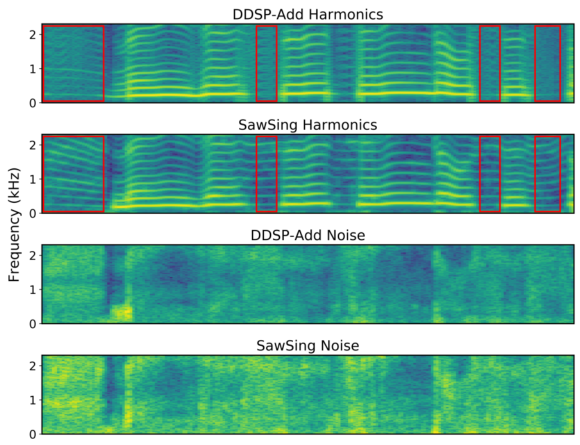


Figure 4: The spectrograms of the harmonic signal x_h and noise signal x_n generated by DDSP-Add and SawSing for the same clip. The red rectangles highlight the moments the buzzing artifact of SawSing emerges.

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

[1] P. R. Cook, "Singing voice synthesis: History, current work, and future directions," *Computer Music Journal*, vol. 20, no. 3, pp. 38–46, 1996.

[2] J. Lee, H.-S. Choi, C.-B. Jeon, J. Koo, and K. Lee, "Adversarially trained end-to-end Korean singing voice synthesis system," in *INTERSPEECH*, 2019.

[3] M. Blaauw and J. Bonada, "Sequence-to-sequence singing synthesis using the feed-forward Transformer," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2020, pp. 7229–7233.

[4] Y. Ren, X. Tan, T. Qin, J. Luan, Z. Zhao, and T.-Y. Liu, "DeepSinger: Singing voice synthesis with data mined from the web," in *ACM Int. Conf. Knowledge Discovery & Data Mining*, 2020, pp. 1979–1989.

[5] J. Chen, X. Tan, J. Luan, T. Qin, and T.-Y. Liu, "HifiSinger: Towards high-fidelity neural singing voice synthesis," *arXiv preprint arXiv:2009.01776*, 2020.

[6] Y. Hono *et al.*, "Sinsy: A deep neural network-based singing voice synthesis system," *IEEE Trans. Audio, Speech and Lang. Proc.*, vol. 29, pp. 2803–2815, 2021.

[7] Y.-P. Cho, F.-R. Yang, Y.-C. Chang, C.-T. Cheng, X.-H. Wang, and Y.-W. Liu, "A survey on recent deep learning-driven singing voice synthesis systems," in *IEEE Int. Conf. Artificial Intelligence and Virtual Reality*, 2021.

[8] C.-F. Liao, J.-Y. Liu, and Y.-H. Yang, "KaraSinger: Score-free singing voice synthesis with VQ-VAE using Mel-spectrograms," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2022.

[9] J. Liu, C. Li, Y. Ren, F. Chen, P. Liu, and Z. Zhao, "DiffSinger: Singing voice synthesis via shallow diffusion mechanism," in *AAAI Conference on Artificial Intelligence*, 2022.

[10] A. v. d. Oord *et al.*, "WaveNet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[11] N. Kalchbrenner *et al.*, "Efficient neural audio synthesis," in *Int. Conf. Machine Learning*, 2018.

[12] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter waveform models for statistical parametric speech synthesis," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 402–415, 2020.

[13] K. Kumar *et al.*, "MelGAN: Generative adversarial networks for conditional waveform synthesis," *arXiv preprint arXiv:1910.06711*, 2019.

[14] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2020, pp. 6199–6203.

[15] J. Kong, J. Kim, and J. Bae, "Hifi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems*, 2020.

[16] Y. Hono *et al.*, "PeriodNet: A non-autoregressive waveform generation model with a structure separating periodic and aperiodic components," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2021.

[17] H. Guo, Z. Zhou, F. Meng, and K. Liu, "Improving adversarial waveform generation based singing voice conversion with harmonic signals," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2022.

[18] F. Chen, R. Huang, C. Cui, Y. Ren, J. Liu, and Z. Zhao, "SingGAN: Generative adversarial network for high-fidelity singing voice generation," in *ACM Multimedia*, 2022.

[19] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "WaveGrad: Estimating gradients for waveform generation," in *Int. Conf. Learning Representations*, 2021.

[20] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A versatile diffusion model for audio synthesis," in *Int. Conf. Learning Representations*, 2021.

[21] R. Huang, M. W. Y. Lam, J. Wang, D. Su, D. Yu, Y. Ren, and Z. Zhao, "FastDiff: A fast conditional diffusion model for high-quality speech synthesis," in *Int. Joint Conf. Artificial Intelligence*, 2022.

[22] M. W. Y. Lam, J. Wang, D. Su, and D. Yu, "BDDM: Bilateral denoising diffusion models for fast and high-quality speech synthesis," in *Int. Conf. Learning Representations*, 2022.

[23] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Int. Conf. Learning Representations*, 2021.

[24] A. Bitton, P. Esling, and T. Harada, "Neural granular sound synthesis," in *Int. Computer Music Conf.*, 2021.

[25] B. Hayes, C. Saitis, and G. Fazekas, "Neural waveshaping synthesis," in *Int. Soc. Music Information Retrieval Conf.*, 2021, pp. 254–261.

[26] N. Masuda and D. Saito, "Synthesizer sound matching with differentiable dsp," *Proc. International Society for Music Information Retrieval*, 2021.

[27] S. Shan, L. Hantrakul, J. Chen, M. Avent, and D. Trevelyan, "Differentiable wavetable synthesis," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2022, pp. 4598–4602.

[28] M. Carney, C. Li, E. Toh, P. Yu, and J. Engel, "Tone Transfer: In-browser interactive neural audio synthesis," in *Workshop on Human-AI Co-Creation with Generative Models*, 2021.

[29] F. Ganis, E. F. Knudesn, S. V. K. Lyster, R. Otterbein, D. Südholt, and C. Erkut, "Real-time timbre transfer and sound synthesis using DDSP," in *Sound and Music Computing Conf.*, 2021.

[30] S. Nercessian, "End-to-end zero-shot voice conversion using a DDSP vocoder," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2021, pp. 1–5.

[31] C.-C. Chu, F.-R. Yang, Y.-J. Lee, Y.-W. Liu, and S.-H. Wu, "MPop600: A Mandarin popular song database with aligned audio, lyrics, and musical scores for singing voice synthesis," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conf.*, 2020, pp. 1647–1652.

[32] J. Sundberg, *The Science of the Singing Voice*. Northern Illinois University Press, 1989.

[33] J. Lane, D. Hoory, E. Martinez, and P. Wang, "Modeling analog synthesis with DSPs," *Computer Music Journal*, vol. 21, no. 4, pp. 32–41, 1997.

[34] A. Huovilainen and V. Välimäki, "New approaches to digital subtractive synthesis," in *Inr. Computer Music Conf.*, 2005.

[35] Z. Liu, K.-T. Chen, and K. Yu, "Neural homomorphic vocoder," in *INTERSPEECH*, 2020.

[36] G. Greshler, T. Shaham, and T. Michaeli, "Catch-a-waveform: Learning to generate audio from a single short example," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[37] S. Nercessian, "Zero-shot singing voice conversion," in *Int. Soc. Music Information Retrieval Conf.*, 2020, pp. 70–76.

[38] K. Ito, "The LJ speech dataset," 2017.

[39] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Int. Conf. Digital Audio Effects*, 2020.

[40] M. A. M. Ramírez, O. Wang, P. Smaragdis, and N. J. Bryan, "Differentiable signal processing with black-box audio effects," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2021, pp. 66–70.

[41] C. J. Steinmetz, J. Pons, S. Pascual, and J. Serrà, "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2021, pp. 71–75.

[42] S. Nercessian, A. Sarroff, and K. J. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2021, pp. 890–894.

[43] B.-Y. Chen, W.-H. Hsu, W.-H. Liao, R. M. A. Martínez, Y. Mitsufuji, and Y.-H. Yang, "Automatic DJ transitions with differentiable audio effects and generative adversarial networks," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2022, pp. 466–470.

[44] J. Alonso and C. Erkut, "Latent space explorations of singing voice synthesis using DDSP," *arXiv preprint arXiv:2103.07197*, 2021.

[45] Y. Wang, X. Wang, P. Zhu, J. Wu, H. Li, H. Xue, Y. Zhang, L. Xie, and M. Bi, "Opencpop: A high-quality open source Chinese popular song corpus for singing voice synthesis," *arXiv preprint arXiv:2201.07429*, 2022.

[46] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[47] 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, pp. 1877–1884, 2016.

[48] A. Gulati *et al.*, "Conformer: Convolution-augmented Transformer for speech recognition," in *INTERSPEECH*, 2020.

[49] K. Kilgour *et al.*, "Fréchet Audio Distance: A metric for evaluating music enhancement algorithms," *arXiv preprint arXiv: 1812.08466*, 2019.

[50] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2018, pp. 161–165.

[51] C. J. Steinmetz and J. D. Reiss, "pyloudnorm: A simple yet flexible loudness meter in Python," in *Proc. AES Convention*, 2021.

[52] Y. Jadoul, B. Thompson, and B. de Boer, "Introducing Parselmouth: A Python interface to Praat," *Journal of Phonetics*, vol. 71, pp. 1–15, 2018.

[53] A. Caillon and P. Esling, "Streamable neural audio synthesis with non-causal convolutions," *arXiv preprint arXiv:2204.07064*, 2022.

[54] E. Deruty, M. Grachten, S. Lattner, J. Nistal, and C. Aouameur, "On the development and practice of AI technology for contemporary popular music production," *Transactions of the International Society for Music Information Retrieval*, vol. 5, no. 1, pp. 35–49, 2022.

EQUIVARIANT SELF-SUPERVISION FOR MUSICAL TEMPO ESTIMATION

Elio Quinton
Universal Music Group

ABSTRACT

Self-supervised methods have emerged as a promising avenue for representation learning in the recent years since they alleviate the need for labeled datasets, which are scarce and expensive to acquire. Contrastive methods are a popular choice for self-supervision in the audio domain, and typically provide a learning signal by forcing the model to be invariant to some transformations of the input. These methods, however, require measures such as negative sampling or some form of regularisation to be taken to prevent the model from collapsing on trivial solutions. In this work, instead of invariance, we propose to use equivariance as a self-supervision signal to learn audio tempo representations from unlabelled data. We derive a simple loss function that prevents the network from collapsing on a trivial solution during training, without requiring any form of regularisation or negative sampling. Our experiments show that it is possible to learn meaningful representations for tempo estimation by solely relying on equivariant self-supervision, achieving performance comparable with supervised methods on several benchmarks. As an added benefit, our method only requires moderate compute resources and therefore remains accessible to a wide research community.

1. INTRODUCTION

Manually annotated data is scarce and expensive to acquire, becoming a bottleneck to the continued progress of supervised learning methods in recent years. This is particularly true in the music domain, where a large proportion of content available is under copyright. Unlabelled data is comparatively abundant and inexpensive. Self-Supervised Learning (SSL) methods, which do not require labeled training data, have shown very promising development by rendering larger unlabelled datasets usable for training and yielding performance comparable or surpassing supervised benchmarks on Natural Language Processing (NLP) [1,2], computer vision tasks [3], speech processing [4] and music tasks [5]. Self-supervised learning is still only nascent in musical audio, in comparison with other domains. To date it has only been considered with target downstream tasks that are a limited subset of the Music Information Retrieval (MIR) field, e.g. auto-tagging, genre classification or cover song detection [5–7]. With this contribution, we propose to extend the application of a SSL framework to the rhythmic properties of music by applying it to the tempo estimation task, which state of the art is still within the realm of supervised methods [8–11].

Self-supervision is typically realised by creating a pretext task providing a training signal from which it is expected that the model can learn useful representations. The general intuition underpinning this family of approaches is that accurately performing on the pretext task requires the model to learn meaningful representations of the domain at hand. For domains where data is made of discrete tokens, such as text, pretext tasks like masked language modelling or next sentence prediction as have proven to be effective and brought a step change in NLP [2]. Taking inspiration from it, comparable approaches can be devised in the symbolic music domain, which data is also in the form of discrete tokens [12]. In domains where data is dense and continuous, such as images or audio, one option is to apply similar token-based methods to discrete representations of the dense input [13, 14]. Alternatively, Siamese network frameworks [15], where two models process two views of a given input, are a popular choice for handling continuous data directly. In particular, contrastive methods where the pretext task consists in discriminating whether two inputs (or set thereof) should yield a similar, or dissimilar representation have been shown to be effective in computer vision [3] and in the audio domain [16–19], including musical audio [5–7].

In the contrastive learning framework, for every training sample, two views are generated by applying random data augmentations. The training objective then constrains the model to produce representations that are *invariant* to the augmentations applied to the training data and yet discriminative between different samples. The choice of augmentations is a critical design choice that impacts the quality and properties of representation learnt [20]. For example, in [5] the model is trained to be invariant to pitch-shifting. While this is appropriate for fine-tuning on an auto-tagging task, it would not be suitable for downstream tasks such as chord or key estimation. Generally speaking, the invariance constraints explicitly specifies what the representations should *not* be sensitive to. Conversely, *equivariant* constraints can be applied to enforce preferences on what features or concepts may be desirable to capture in the learnt representations [21, 22]. This is particularly attractive in scenarios where it is not clear how invariance constraints can yield a suitable representation, such as F0 estimation [23], or for guaranteeing that the representations capture specific semantically meaningful dimensions such as harmony or rhythm, which may be beneficial for applications such as music search and discovery.

In this work, instead of *invariance*, we propose to use *equivariance* as a self-supervision constraint to learn audio representations that specifically capture musical tempo, from unlabelled data. We propose a Siamese network framework where we produce two views of each training sample by applying a time stretching transformation with random factor and impose an equivariance constraint between the representations. Although the tempo of the training samples is unknown, as a result of the time-stretching transformation, the tempo of each sample is modified in two different, but known, ways. From there we derive a loss function that exploits this information and enforces the equivariant constraint. Because we wish the representations to capture rhythmic properties of music only, we also add audio augmentations to promote robustness against potentially confounding attributes of the audio signal. As is customary with self-supervised pre-training [3, 5], we evaluate the quality of the representations learnt with our method by freezing the model and fine tuning a linear layer on a downstream tempo estimation task.

Our key contributions are the following. 1- To the best of our knowledge, our work is the first to rely solely on an equivariance-based objective to learn musical audio representations. 2- We derive a simple loss function that formally prevents the network from collapsing on a trivial constant solution during training, without requiring any form of regularisation or negative sampling. 3- Our experiments show that it is possible to learn meaningful representations for tempo estimation by solely relying on equivariant self-supervision, and to achieve performance comparable with supervised methods on several benchmarks. 4- Our experiments demonstrate out-of-domain transferability of the representations learnt across multiple datasets. 5- As an added benefit, our method only requires moderate compute resources and therefore remains accessible to a wide research community. In order to further reproducibility we make code and pre-trained models available [1].

2. BACKGROUND

2.1 Tempo Estimation

In many musical traditions, tempo is one of the fundamental characteristics of music. Tempo estimation was also one of the first tasks to have been explored in the field of MIR [24–28]. Historically, tempo estimation was performed by first extracting an onset detection function [29,30] and then perform tempo estimation via the computation of inter-onset intervals or some form of periodicity function from the onset detection curve [24,26,28,31,32].

With the introduction of deep learning, like most other MIR tasks, tempo estimation methods have gradually moved towards end-to-end learning where the deep neural network would jointly perform the feature extraction (e.g. onset detection curve) and tempo estimation [8–10]. This evolution brought a boost in performance so that the tempo estimation state of the art benchmark have been based on supervised deep learning for some years [8–10]. In line with this observation, we also adopt a deep neural network model described in detail in Section 3.1.

2.2 Siamese networks and trivial solutions

Most of the methods that have been shown to be successful at self-supervised representation learning for domains where data is dense and continuous, such as image and audio [3, 5, 33–35], are variations of the Siamese networks architecture [15]. Another trait these methods have in common is that they aim to maximise the similarity between the representations obtained from different transformations of a training sample. This problem, however, admits trivial solutions. For example, if the model produces a constant output irrespective of input, the representations of the two views would always be identical. Providing a mitigation strategy to prevent the model to collapse to a trivial solution is a major challenge in this family of methods.

A variety of strategies have been proposed to guard against the trivial solution collapse issue in the literature. SimCLR relies on negative sampling, where for each training sample all other samples in the mini-batch are considered as negatives [3]. This class of methods tend to benefit from large batch sizes. The Barlow twins approach prevents collapse by introducing an additional redundancy term to the training loss [36]. The authors also note that the method require smaller batch size than SimCLR. Deep cluster [37] and SwAV [33] prevents collapse by incorporating a clustering mechanism so that image features are not compared directly. Departing from the contrastive approach, BYOL relies only on positive pairs and prevents collapse by introducing asymmetry in the training scheme by updating the model parameters using only one transformed view of the input while the other view is used as a target [34]. In a similar line of work, the SimSiam authors note that the 'stop-gradient' operation introduces some asymmetry that is critical in preventing the collapse to trivial solutions [38]. Also relying on asymmetry, MoCo mitigates the trivial solution collapse by including momentum encoder [39].

The methods described above introduce trivial solution collapse mitigation strategies of varying degree of complexity that are found to work in practice even though their formulation still formally accepts trivial constant solutions. In Section 3.2 we propose a simple framework that formally does not admit a trivial constant solution, does not require negative samples [3,5], large batches [3,5], asymmetry [34,35,38,39], non-differentiable operators [33] or stop gradients [38].

2.3 Equivariant objective

A representation $q(w)$ is invariant if it remains unchanged for a certain transformation k of the input w:

[1] https://github.com/Quint-e/equivariant-self-supervision-tempo