Speech-to-Singing Conversion based on Boundary Equilibrium GAN

Da-Yi Wu[1,2] *and Yi-Hsuan Yang*[2]

[1]Department of CSIE, National Taiwan University
[2]Research Center for IT Innovation, Academia Sincia
r07922119@ntu.edu.tw, yang@citi.sinica.edu.tw

Abstract

This paper investigates the use of generative adversarial network (GAN)-based models for converting a speech signal into a singing one, without reference to the phoneme sequence underlying the speech. This is achieved by viewing speech-to-singing conversion as a style transfer problem. Specifically, given a speech input, and the F0 contour of the target singing output, the proposed model generates the spectrogram of a singing signal with a progressive-growing encoder/decoder architecture. Moreover, the model uses a boundary equilibrium GAN loss term such that it can learn from both paired and unpaired data. The spectrogram is finally converted into wave with a separate GAN-based vocoder. Our quantitative and qualitative analysis show that the proposed model generates singing voices with much higher naturalness than an existing non adversarially-trained baseline.
Index Terms: Speech-to-singing conversion, singing voice synthesis, style transfer, adversarial training, encoder/decoder.

1. Introduction

The goal of singing voice synthesis is to create natural-sounding singing voices with some given conditions, such as the lyrics, pitch labels, or reference audio [1, 2, 3, 4, 5]. The reference audio can be a singing passage of a person, and the task is to convert the timbre of the singing passage into the timbre of someone else [6]. The reference audio can also be a passage of speech voice by someone, and the task is to convert it into a singing passage with the same timbre identity and linguistic content, without reference to the underlying phoneme sequence [7, 8, 9]. We are interested in the latter task, i.e., *speech-to-singing* (STS) conversion, in this paper, for its interesting applications in entertainment, karaoke, music production, and others.

Even through there are many properties shared by speech and singing signals, they cannot be easily converted to one another. Rhythm and phoneme representations in speech and singing are fairly different, and one singing sound can correspond to spoken passages with different speeds, tones, and even pronunciations [8]. Moreover, melody information, less important for speech, is critical and indispensable for the expression of singing. Hence, in addition to preserving the linguistic content and timbre identity, an STS model would also need to generate singing that follows a pre-given or automatically-generated melody contour.

In the literature, there have been two main approaches to STS: model-based and template-based ones. For **model-based STS**, the work presented by Saitou *et al.* [7] is a representative one. They decompose the STS process into three main parts and process the signal using different control models. This involves lengthening the speech by a duration control model, generating the F0 contour with an F0 model, and modifying the timbre of the voice (so that it is singing-like) by a spectral model. Yet, the synthesis quality much depends on how accurate the phonemes are segmented and associated with the musical notes.

For **template-based STS**, proposed for the first time in [10], assumes that a high quality-singing vocal, a.k.a., the "template singing," is available as another audio reference. The inputs therefore comprise the speech and the template singing, which are to be firstly aligned with one another. The template singing is further used to extract the reference prosody which includes singing F0, aperiodicity index, singing formants, etc. This information is then used to estimate the parameters of singing synthesis from the aligned speech. As another example, Gao *et al.* [11] propose a deep learning based model for template-based STS conversion which conditions the network on the i-vector of a speaker while predicting the singing spectral parameters to preserve the speaker identity.

In this paper, we view STS as a **style transfer** problem, which can be considered as the third approach to STS. Speech and singing are two very different styles, in that in singing we care more about melody and rhythm. Among the existing style transfer architectures, we adopt the GAN [12] architecture for its generalizability, and demonstrated effectiveness in other musical style transfer tasks [6, 13].

This work represents a continuation and extension of our prior work presented in [9], which is among the first attempts to approach STS with a deep-learning model that does not require any phoneme synchronization information or high-quality singing reference. In other words, only a speech passage and a target melody (F0) contour is needed for this model. We show in [9] that even there is no other additional information, the model can still learn to sing, albeit the quality of synthesis is not good enough. The limited quality of the generated singing can be attributed to many parts of the model, and it is our goal to improve upon this prior art here by introducing adversarial learning and modern style transfer techniques.

Specifically, our contributions are as follows. First, we replace the straightforward convolution architecture of [9] by an architecture that progressively grows the output spectrogram with a hierarchical structure. Such an idea has been increasingly used to generate audio in recent work [14, 15, 16, 17]. Second, instead of relying fully on paired data as in [9], we extend the model so that it can also learn from unpaired data. Specifically, an adversarial architecture based on boundary-equilibrium GAN [18] is employed. Third, we adopt the 'random resampling' idea proposed in the autoVC paper [19] to better disentangle content and rhythm information. Finally, we use a MelGAN-based neural vocoder [20] to replace the Griffin-Lim algorithm [21] adopted in [9] to further improve the sound quality.

For paired data, we use the the NUS sung and spoken lyrics corpus [22],[1] which is the largest public paired dataset to date. For unpaired data, we use the DAMP corpus [23],[2] which is comprised of performances of 5,690 unique popular songs.

[1]https://smcnus.comp.nus.edu.sg/nus-48e-sung-and-spoken-lyrics-corpus/
[2]https://ccrma.stanford.edu/damp/

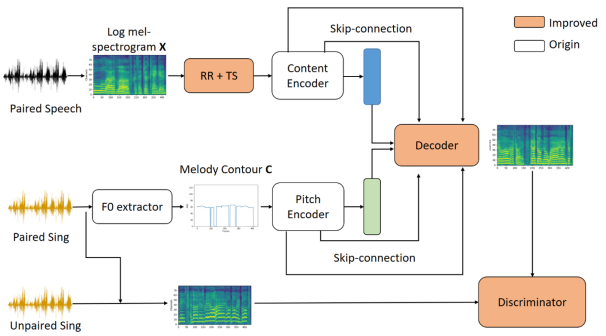

Figure 1: *Diagram of the proposed model architecture, which is extended from the model we previously presented in [9]. The input audio is represented by a log mel-spectrogram, and the target melody contour is represented by the one-hot format. The model learns to perform STS conversion by joint supervised and unsupervised learning. We highlight the components that are different from the prior art [9] in orange. 'RR' stands for random resampling [19], while 'TS' stands for time stretching.*

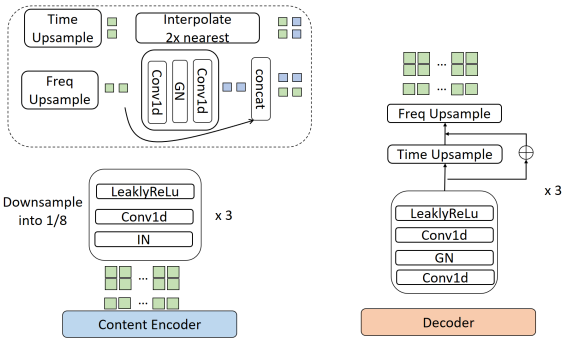

Figure 2: *Details of the employed encoder and decoder. The encoder consists of convolution banks, and the decoder upsamples the time-frequency representation by interpolation and concatenation. 'IN' and 'GN' represent instance normalization and group normalization, respectively. Details of time and frequency upsample blocks are pictured on the top-left side.*

We report both quantitative and qualitative evaluation to compare the proposed model with the prior art [9] as well as ablated versions of our model. Open source implementation of the proposed model, as well as audio examples of the generated conversions can be found online.[3,4]

2. Methods

We perform spectra-to-spectra conversion in an encoder-decoder framework as depicted in Fig. 1. The input speech is transformed into a log mel-spectrogram, and the F0 contour is extracted by a vocal melody extractor from a different source such as humming or reference singing. We time-stretch the mel-spectrogram of speech to the same length as its target F0 contour, and apply two encoders to encode speech and pitch information separately. Then, the decoder takes the concatenation of these two encodings and generates the singing output, with skip-connections from the encoder. Finally, we use MelGAN vocoder to reconstruct the waveform from the log mel-spectrogram. Additionally, we add BEGAN discriminator trained on both model output and ground truth to improve the model generalizability. The whole model is trained on both paired and unpaired data. We explain below each of the components in more detail.

2.1. Input processing

Given an input time-domain speech signal and a target F0 contour, the pre-processing consists of the following steps.

Log-magnitude representation: We compute the magnitude of mel-spectrogram for the speech signal and apply element-wise logarithm transformation, yielding a matrix $\mathbf{X} \in \mathbb{R}^{F \times T}$, with F frequency bins and T time frames.

Random resampling (RR): A speech passage can be "sung" with various speed and rhythmic characteristics. Following autoVC [19], we random resample the input speech to change the rhythm of the input speech, so as to better disentangle content and rhythm. This is done by firstly dividing the mel-spectrogram of speech into segments of random lengths, and then randomly stretching or squeezing each segment along the time axis. In our implementation, we divide the speech into segments of random length of 16–32 frames, and temporally stretch each segment by a factor of 0.5–2.

Singing melody contour is extracted from a singing audio using CREPE [24], a state-of-the-art, open source monophonic pitch tracker. Moreover, we convert the continuous-valued extracted F0 to one of the 128 MIDI notes by referencing to the Hz-to-MIDI function in the librosa package [25], and convert it into the one-hot format. As a result, the melody contour $\boldsymbol{C}$ is represented as a sequence of one-hot vector indicating the target MIDI note per frame, namely $\boldsymbol{C} \in N^{128 \times T'}$. Following [9], in the case of training with paired data, we extract the melody contour from the singing counterpart of the input speech.

Time stretching (TS). The speech mel-spectrogram and the target F0 contour (which is extracted from a singing signal) are usually much different in length. Therefore, it would be better if our model does not assume that the input and output are of the same length. However, existing models addressing the variable-length task, such as the sequence-to-sequence model Tacotron [26], often deal with only one of the following two cases: 1) both the input and output are discrete, or 2) the input is discrete while the output is continuous. Spectra-to-spectra learning tasks like STS have to deal with the case where both the input and output are continuous, something that has not been widely studied yet. Therefore, we use a simpler **fixed-length setting** (and left the variable-length version as a future work). Accordingly, the input mel-spectorgram is linearly interpolated to $\mathbf{X} \in \mathbb{R}^{F \times T'}$, i.e., to be of the same length as the F0 contour.

2.2. Encoder and Decoder

As shown in Fig. 2, the content encoder uses 3 stacks of layers to obtain the latent code, downsampling both time and frequency by a factor of 8. Each stack is composed of a instance normalization layer [27] (as a "style remover"), a 1D-convolution layer (with 3×1 convolution kernels with stride 2; no pooling), and a LeakyReLu activation layer. On the other hand, the pitch encoder (not shown in Fig. 2) applies embedding layer to project the one-hot vectors $\boldsymbol{C}$ to an embedding space first, and then passes it through similar 1D-convolution based layers.

As for the decoder, we use a progressive-growing architecture, which has been recently shown to be an effective "spectro-

[3] https://github.com/ericwudayi/speech2singing

[4] https://ericwudayi.github.io/Speech2Singing-DEMO

gram generator" recently [15, 16]. Specifically, it uses 3 stacks of layers, each stack comprising two 1D 3×1 kernel convolutions with stride 1, a Group-Norm layer [28], and two upsampling modules. Following UNAGAN [16], nearest neighbor interpolation is applied instead of 1D transposed-convolution. And, Upsampling in frequency domain is performed by concatenation instead of a convolution layer, in the similar light of [15, 16].

2.3. BEGAN

BEGAN is an energy-based GAN architecture [18]. While the original GAN [29] matches the distributions between the real and generated samples directly, an energy-based GAN matches the distribution of loss using an auto-encoder architecture. Moreover, BEGAN relaxes the equilibrium of the auto-encoder loss using a hyper-parameter $\gamma \in [0, 1]$ (a.k.a, diversity ratio), defined as

$$\gamma = \mathbb{E}[L(G(\mathbf{x}))]/\mathbb{E}[L(\mathbf{y})]\,, \tag{1}$$

where $G(\mathbf{x})$ is a generated sample, $\mathbf{y}$ is a real sample, and $L(\cdot)$ is the reconstruction loss function of the auto-encoder. The γ term balances the diversity and quality. Lower values of γ make the discriminator focuses more on real samples, generating samples with better quality but less diversity. In contrast, higher values of γ improve the diversity but lower the quality.

We use BEGAN as our unpaired data trainer for its stable training process observed in our pilot study. It has also been shown in [30] and [31] that BEGAN performs better than some other commonly-used GAN models for generating audio.

2.4. Training

Our model is trained with by minimizing the BEGAN loss and L1 loss jointly. Formally, given an input log mel-spectrogram $\mathbf{X}$, a melody contour $\mathbf{C}$, and a target singing log mel-spectrogram $\mathbf{Y}$, the losses of the generator and discriminator are defined as:

$$\begin{cases} L_D = L(\mathbf{Y}) - k_t L(G(\mathbf{X}, \mathbf{C}))\,, \\ L_G = L(G(\mathbf{X}, \mathbf{C})) + \beta(|\mathbf{Y} - G(\mathbf{X}, \mathbf{C})|_1)\,, \end{cases} \tag{2}$$

where L_D only depends on BEGAN loss, and the variable $k_t \in [0, 1]$ controls how much emphasis is put on $L(G(x))$. The update of k_t is controlled by the diversity ratio γ according to

$$k_{t+1} = k_t + \lambda(\gamma L(\mathbf{Y}) - L(G(\mathbf{X}, \mathbf{C})))\,. \tag{3}$$

In Eqn. (2), we also use the pixel-wise $L1$ spectrogram loss to L_G to achieve supervised learning in paired data, scaling with a β factor to avoid overfitting. As for the case of training with unpaired data, the generator is trained with $L(G(\mathbf{x}))$ only.

2.5. Vocoder

Neural vocoders such as the WaveNet vocoder [32] have been shown to outperform the traditional Griffin-Lim algorithm [21] for converting time-frequency representations to waveforms. Among the neural vocoders that have been proposed, we adopt MelGAN [20] for its remarkable efficiency and generalizability.

3. Experimental Validation

3.1. Setup

Training and test sample generation. We employ the NUS database [22] as our paired training dataset, which contains 115 mins of singing data and the corresponding 54 mins of speech data. 48 recordings of 20 unique English songs are sung and read out by 12 subjects, and each sung-read paired audio can be time-aligned by their phone-duration annotations. The phone annotations is in accordance to the CMU phoneme dictionary (39 phonemes)[5] with two extra phones denoting the silence and inhalation between words, and the duration annotations specify the start and end time of each phone. We remove the silence from speech by using the phoneme-duration annotation. All the speech input are constrained to have three or more words.

Out of the 20 unique songs in the dataset, we keep one song (with two recordings) as our test set. The test singer is only present in one recording in our training set, and the test song is not seen at all in the training data.

Due to the small size of the NUS dataset, we perform data augmentation using unpaired data with the DAMP dataset [23], a multi-singer, singing-only dataset comprising of performances of 5,690 unique songs by 13,154 users of a karaoke App (i.e. a song can be sung by different people). We exclude the segments of song that contains silences of longer than one second.[6]

Implementation details. All the audio processing procedures are performed using librosa [25]. The time domain signals are resampled from 44k to 22k Hz. We compute STFT with 1,024-pt FFT size, 12.5 milliseconds hop size, and transform it into 80-bin mel-scale. We set BEGAN parameters $\lambda = 0.01$ and initialize $k_0 = 0$, $\gamma = 0$. We set supervised learning factor $\beta = 0.5$ and use Adam as our optimizer with initial learning rate 0.001 and exponential decrease factor of 0.99. The neural network is implemented using pytorch. We train the networks for 20k steps with 32 batch size. The training process takes about 5 hours to complete on an NVIDIA RTX 1080-Ti GPU.

Our multi-singer MelGAN vocoder is trained on the union of the NUS dataset [22] and two other sources of unaccompanied singing data, the DSD [34] and MUSDB18 datasets [35].[7] The whole training process take 1000k steps, for about 2 weeks on the same 1080-Ti GPU.

Phoneme synchronization (PhSync). To quantify how much the burden of modelling phoneme duration affects STS performance, as an *oracle* method we follow the settings in [9] and stretch each phone in the input speech to be the same length as the target singing. The duration for each phoneme is obtained from the phone-level annotations for speech and singing.

3.2. Models evaluated

We evaluate several ablated versions of our model to study the effect of the proposed changes.

- *Baseline*: Uses the model architecture proposed in the prior art [9], which is trained through multi-task learning with phoneme prediction, using only the paired data. It uses log spectrograms as its input, not mel-spectrograms.
- *Decoder* **(D)**: Uses the modified version of the decoder, and trained with the MSE loss. We use the spectrograms here as the input for fair comparison with the Baseline.

[5] `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

[6] Our originally plan is to also use LJSpeech [33], a single-speaker speech-only dataset with 24 hours of reading audio for data augmentation. We later realize that this is not easy, as in our STS setting, we need to find a way to create the target melody contour for each input speech. The melody contour should correlate well to the number of words from the spoken lines. And, it is not natural to convert the speech with arbitrary melody contour. We attempt to give the pitch contour by randomly picking a pitch contour from DAMP, but the result shows that it actually hurts the audio quality. We therefore finally decide not to use LJSpeech.

[7] These two datasets provide unaccompanied singing so there is no need to perform singing/accompaniment source separation [36].

Model	LSD ↓	LSD (mel) ↓	RCA ↑
Baseline [9]	9.97	—	0.760
D	9.36	—	0.801
D+A	**9.21**	—	**0.820**
D+V	—	1.15	0.811
D+V+A	—	**1.13**	**0.832**
D+V+A + PhSync	—	1.07	0.816

Table 1: *Results on objective metrics for different proposed STS models. Log-spectral distance (LSD; in db) is the lower the better, while raw chroma accuracy (RCA) the inverse. The first three models take spectrogram as input and use Griffin-Lim for inverse STFT, while the others use mel-spectrorams and MelGAN-based vocoder. Therefore, 'LSD' and 'LSD (mel)' are calculated over magnitude spectrograms and mel spectrograms, respectively. We highlight the best result obtained, excluding the last model as it uses oracle phone-annotation (PhSync) information.*

- *Decoder + Adversarial* **(D+A)**: Uses the modified version of the decoder, and trained with MSE and adversarial losses jointly. Also use the spectrogram features. In other words, the unpaired data is used.
- *Decoder + Neural Vocoder* **(D+V)**: Uses our modified version of the decoder, and trained instead on the log mel-spectrogram features. The output is converted to waveform by MelGAN [20] instead of Griffin-Lim.
- *Complete model* **(D+V+A)**: The proposed model trained with adversarial loss and MSE loss jointly using both paired and unpaired data, and the output is also converted into waveform by MelGAN.
- *Complete model + PhSync*: The proposed model trained in the oracle phoneme synchronization setting.

3.3. Objective Evaluation

Following [9], we use log-spectral distance (LSD) and F0 raw chroma accuracy (RCA) [37] as the objective metrics. LSD evaluates the dissimilarity between two spectra, while RCA evaluates melody correctness. LSD is computed by averaging the Euclidean distance between true and predicted log spectrogram or log mel-spectrogram frames over time, for frequencies between 100–3500 kHz. RCA is computed between a target waveform and the model output, and we set the maximum tolerance deviation between target and output as 50 cents [37]. The models are evaluated by selecting random 50 test samples with speech duration of at least 2 seconds and then averaging the scores over all the samples.

The result is shown in Table 1. Several observations can be made. First, **D** performs noticeably better than Baseline on both LSD and RCA. This indicates that our decoder architecture combined with the progressive-growing architecture works better for STS. Second, the models with adversarial learning consistently outperform the non-adversarial counterparts (i.e., **D+A** outperforms **D**, while **D+V+A** outperforms **D+V**). This suggests that unsupervised learning with the BEGAN architecture improves the result. Finally, in terms of RCA, the best result (0.832) is obtained by the complete model (**D+V+A**).[8]

[8]We note that the first three methods take the magnitude spectrograms as inputs, while the other methods deal with the mel-specgrograms. Therefore, the values of 'LSD' and 'LSD (mel)' are not comparable.

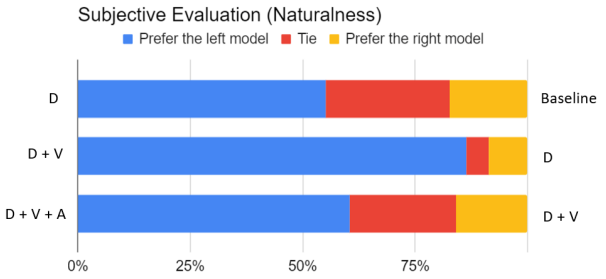

Figure 3: *Subjective evaluation results for pairs of systems, from top to bottom: 'D vs. Baseline,' 'D+V vs. D,' 'D+V+A vs. D+V.'*

3.4. Subjective Evaluation

Instead of doing subjective evaluations on all the models, we conducted preference listening test on the following three selective pairs of models: '**D** vs. Baseline,' '**D+V** vs. **D**,' '**D+V+A** vs. **D+V**,' to respectively investigate: 1) Is our decoder architecture better than the prior one employed in [9]? 2) Does the neural vocoder with the 80-bin mel-spectrogram performs better than the Griffin-Lim algorithm with the 512-bin spectrogram? 3) Does adversarial learning help generate more natural-sounding singing? Each participant was first asked to listen the input speech and then compare the outputs of a given pair of models. The participants were then asked to specify their preference among the two models in terms of naturalness. For all pairs of models, the percentage of votes (including ties) are reported.

The response from 38 participants recruited from the Internet for our listening test is summarized in Fig. 3. Each of our modifications seems to outperform the corresponding ablated version. Key takeaways are: We can do better STS by using a neural vocoder on mel-spectrograms, and the use of adversarial learning and unpaired data is beneficial.

3.5. Other Experiments and Future works

As the target melody may not be always available, it would be great if the STS model can generate the melody contour on its own given the speech audio. Hence, in a pilot study we attempt to directly transform speech to singing without prespecified melody. The cycle-BEGAN architecture is employed to achieve unsupervised learning, and models are trained with adversarial loss and cycle reconstruction loss jointly. However, our preliminary result is not positive; our model does not learn to generate coherent melody across time. We plan to study in our future work a two-step model that generates the melody contour from speech first, and then generates the singing. Another idea is to use an auto-regressive models such as the Transformers [38, 39, 40] so that the output is conditioned on previous states.

4. Conclusions

In this paper, we have presented various modifications that improve an end-to-end speech-to-singing conversion neural net. The model takes only speech and target melody contour as the input. It learns from both paired and unpaired data to perform the conversion. We have also report objective and subjective evaluations to validate the effectiveness of the proposed improvements. In the future, we plan to employ sequence-to-sequence models to tackle the task in the variable-length setting, preferably without pre-specified singing melody contours.

5. References

[1] K. Saino, H. Zen, Y. Nankaku, A. Lee, and K. Tokuda, "An HMM-based singing voice synthesis system," in *Proc. Int. Conf. Spoken Language Processing*, 2006.

[2] R. Valle, J. Li, R. Prenger, and B. Catanzaro, "Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2020.

[3] K. Nakamura, K. Hashimoto, K. Oura, and K. T. Yoshihiko Nankaku, "Singing voice synthesis based on convolutional neural networks," *arXiv preprint arXiv:1904.06868*, 2019.

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

[5] Y. Gu, X. Yin, Y. Rao, Y. Wan, B. Tang, Y. Zhang, J. Chen, Y. Wang, and Z. Ma, "ByteSing: A Chinese singing voice synthesis system using duration allocated encoder-decoder acoustic models and wavernn vocoders," *arXiv preprint arXiv:2004.11012*, 2020.

[6] C.-W. Wu, J.-Y. Liu, Y.-H. Yang, and J.-S. R. Jang, "Singing style transfer using cycle-consistent boundary equilibrium generative adversarial networks," in *Proc. Joint Workshop on Machine Learning for Music*, 2018.

[7] T. Saitou, M. Goto, M. Unoki, and M. Akagi, "Speech-to-singing synthesis: Converting speaking voices to singing voices by controlling acoustic features unique to singing voices," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2007, pp. 215–218.

[8] K. Vijayan, H. Li, and T. Toda, "Speech-to-singing voice conversion: The challenges and strategies for improving vocal conversion processes," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 95–102, 2019.

[9] J. Parekh, P. Rao, and Y.-H. Yang, "Speech-to-singing conversion in an encoder-decoder framework," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2020.

[10] L. Cen, M. Dong, and P. Chan, "Template-based personalized singing voice synthesis," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2012, pp. 4509–4512.

[11] X. Gao, X. Tian, R. K. Das, Y. Zhou, and H. Li, "Speaker-independent spectral mapping for speech-to-singing conversion," in *Proc. APSIPA ASC*, 2019.

[12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Computer Vision*, 2017.

[13] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A universal music translation network," *arXiv preprint arXiv:1805.07848*, 2018.

[14] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. Int. Conf. Learning Representations*, 2018.

[15] S. Vasquez and M. Lewis, "Melnet: A generative model for audio in the frequency domain," *arXiv preprint arXiv:1906.01083*, 2019.

[16] J.-Y. Liu, Y.-H. Chen, Y.-C. Yeh, and Y.-H. Yang, "Unconditional audio generation with generative adversarial networks and cycle regularization," in *Proc. INTERSPEECH*, 2020.

[17] M. Bińkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan, "High fidelity speech synthesis with adversarial networks," in *Proc. Int. Conf. Learning Representations*, 2020.

[18] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[19] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, "AUTOVC: Zero-shot voice style transfer with only autoencoder loss," in *Proc. Int. Conf. Machine Learning*, 2019.

[20] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Zhen, T. J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "MelGAN: Generative adversarial networks for conditional waveform synthesis," in *Proc. Annual Conf. Neural Information Processing Systems*, 2019, pp. 14 910–14 921.

[21] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[22] Z. Duan, H. Fang, B. Li, K. C. Sim, and Y. Wang, "The NUS sung and spoken lyrics corpus: A quantitative comparison of singing and speech," in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conf.*, 2013, pp. 1–9.

[23] J. C. Smith, "Correlation analyses of encoded music performance," *Ph.D. Thesis, Stanford University*, 2013.

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

[25] B. McFee, C. Raffel, D. Liang, D. P. W. . Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in Python," in *Proc. Python in Science Conf.*, 2015, pp. 18–25, [Online] https://librosa.github.io/librosa/.

[26] J. Shen *et al.*, "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2017, pp. 4779–4783.

[27] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[28] Y. Wu and K. He, "Group normalization," in *Proc. European Conf. Computer Vision*, 2018, pp. 3–19.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[30] C.-W. Wu, J.-Y. Liu, Y.-H. Yang, and J.-S. R. Jang, "Singing style transfer using cycle-consistent boundary equilibrium generative adversarial networks," in *Proc. Joint Workshop on Machine Learning for Music, extended abstract*, 2018.

[31] J.-Y. Liu, Y.-H. Chen, Y.-C. Yeh, and Y.-H. Yang, "Score and lyrics-free singing voice generation," in *Proc. Int. Conf. Computational Creativity*, 2020.

[32] A. v. d. 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," *arXiv preprint arXiv:1609.03499*, 2016.

[33] K. Ito, "The LJ speech dataset," 2017. [Online]. Available: https://keithito.com/LJ-Speech-Dataset/

[34] A. Liutkus *et al.*, "The 2016 signal separation evaluation campaign," in *Proc. Int. Conf. Latent Variable Analysis and Signal Separation*, 2017, pp. 323–332.

[35] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[36] J.-Y. Liu and Y.-H. Yang, "Dilated convolution with dilated gru for music source separation," in *Proc. Int. Joint Conf. Artificial Intelligence*, 2019, pp. 4718–4724.

[37] C. Raffel, B. McFee, E. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. Raffel, "mir_eval: A transparent implementation of common MIR metrics," in *Proc. Int. Society for Music Information Retrieval Conf.*, 2014.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[39] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based modeling and generation of expressive Pop piano compositions," in *Proc. ACM Int. Conf. Multimedia*, 2020.

[40] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.