INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Unsupervised Cross-Domain Singing Voice Conversion

Adam Polyak[1,2]*, *Lior Wolf*[1,2], *Yossi Adi*[1], *Yaniv Taigman*[1]

[1]Facebook AI Research
[2]The School of Computer Science, Tel Aviv University

adampolyak@fb.com

Abstract

We present a wav-to-wav generative model for the task of singing voice conversion from any identity. Our method utilizes both an acoustic model, trained for the task of automatic speech recognition, together with melody extracted features to drive a waveform-based generator. The proposed generative architecture is invariant to the speaker's identity and can be trained to generate target singers from unlabeled training data, using either speech or singing sources. The model is optimized in an end-to-end fashion without any manual supervision, such as lyrics, musical notes or parallel samples. The proposed approach is fully-convolutional and can generate audio in real-time. Experiments show that our method significantly outperforms the baseline methods while generating convincingly better audio samples than alternative attempts.

1. Introduction

We tackle the audio synthesis task of singing voice conversion. In this task, a given template song is reproduced by another singer's voice. The conversion retains the content and musical expression of the template song. Singing voice conversion can aid in improving the vocal qualities of a given singing segment, create mimicry effects and even enable a single amateur singer to record an entire chorus in their home studio.

Our method is inspired by recent work on speech voice conversion [1] and music synthesis [2]. We combine melody extracted features with acoustic speech features and a powerful neural audio generation framework [3] to create a singing voice synthesizer. All features are extracted directly from the raw audio, therefore, our method does not require supervision in the form of a labelled dataset with lyrics and notes or a parallel dataset with singers singing the same songs.

From a technical perspective, we present multiple contributions: (i) introducing an audio generation framework which employs task-related perceptual losses to improve the quality of the generated audio, (ii) the first singing voice conversion method, as far as we know, conditioned on sine-excitation of the song's melody, together with intermediate features of a speech recognition network, and (iii) presenting a speaker invariant singing voice conversion method trained on voices from either speech or singing datasets (i.e., mimicking in singing either speaking or singing voices), with either single or multiple identities.

2. Related work

Neural Audio Generation Recent advancements in neural audio generation enabled computers to generate natural sounding speech and music. Autoregressive models, such as WaveNet [4] and SampleRNN [5] generate high-quality audio in the waveform domain, one sample at a time, resulting in slow inference. WaveRNN [6] enabled fast synthesis by training a compact neural network further optimized via sparsification. Feed-forward networks were suggested to further speed up inference, via knowledge distillation from an autoregressive teacher-model [7–9]. WaveGlow [10] trained a feed-forward network without knowledge distillation. Recently, generative adversarial networks (GANs) [3, 11] were able to match the quality of autoregressive and large feed-forward models. ParallelWaveGAN [3] trained a mel-inversion network, by combining adversarial networks with a multi-scale spectral loss. Other methods, employed multi-scale spectral loss to train convolutional neural networks for spectrogram inversion [12], to directly predict the parameters of a differentiable synthesizer [13] and for speech synthesis with sine excitation as input [14].

Singing Synthesis and Conversion Previous singing synthesis methods applied unit selection methods [15] or HMM based parametric methods [16–18]. Neural networks were used to synthesize singing [19], by training a WaveNet-like network conditioned on notes and lyrics to generate vocoder features. This method was later extended to synthesize new singers based on a few minutes of them singing [20]. Mellotron [21], presented a sequence to sequence architecture conditioned on text and pitch to synthesize singing, without training on a singing dataset. Recently, non-autoregressive models were used for singing synthesis. Feed-forward transformers [22] removed the constraint of time-aligned phonemes, by employing a duration model. Generative adversarial networks [23], conditioned on pitch and time-aligned phonemes, predicted a singing spectrogram in a single pass instead of a frame-by-frame prediction.

Initial methods for the task of singing voice conversion [24–26], relied on parallel datasets composed of paired samples of singers singing the same piece. The method of Unsupervised Singing Voice Conversion [27] learned to convert between a fixed set of singers without relying on a parallel-dataset. This was done by learning singer-agnostic features via a domain confusion term on the singer identity. PitchNet [28] further improved the method, by applying an additional domain confusion term on the pitch, but was only applied on conversion between a fixed group of male singers.

Very recently, variational autoencoders were used to convert between singers and their vocal techniques [29], learned from non-parallel corpora. However, their method was limited to mel-spectrograms, thereby bounding the audio quality. A method employing an automatic speech recognition (ASR) engine [30] was used for singing voice conversion. The ASR extracts phonemes probabilities, which the model then converts to the acoustic features of the target singer. The method was demonstrated on a many-to-one scenario and was only trained on a singing dataset, while our method is applicable both to many-to-many conversion and can also be trained on either speech or singing domains. Similar to us, [31] used WaveRNN conditioned on phoneme probabilities, pitch and a speaker i-vector to generate singing audio in the waveform-domain. Our

*The contribution of Adam Polyak is part of a Ph.D. thesis research conducted at Tel Aviv University.

http://dx.doi.org/10.21437/Interspeech.2020-1862

method differs by (i) using a non-autoregressive model for real-time generation, and (ii) the usage of perceptual losses which, as we demonstrate, greatly boost the performance of our method.

3. Method

The proposed model is based on a Generative Adversarial Network, with a generator network G and a discriminator network D. The model is conditioned on both speech and musical features, while in the multi-singer generation case, it is additionally conditioned on a learned singer identity vector. Each feature set is forwarded via a separate context-stack, similar to [2, 32], before feeding it to G. The generator is a non-autoregressive WaveNet, which generates the audio waveform directly from a random noise vector. Fig. 1 depicts the architecture.

Input Features We denote the domain of audio samples by $\mathcal{X} \subset \mathbb{R}$. The representation for a raw speech signal is therefore a sequence of samples $\boldsymbol{x} = (x_1, \ldots, x_T)$, where $x_t \in \mathcal{X}$ for all $1 \leq t \leq T$. The length of the input signal varies for different inputs, thus the number of input samples in the sequence, T, is not fixed. Given a training set of n examples, $S = \{\boldsymbol{x}_i\}_{i=1}^{n}$, we would like to extract representations which are both speaker invariant, to enable better singer conversion, and independent of manual annotations, to utilize unlabelled data.

Recent works demonstrated the need of both linguistic and musical features [19, 20, 30] in the context of singing generation. As a result, we extract both the loudness measure [33] and the fundamental frequency (F0) as the musical features. Loudness is represented by the log-scaled A-Weighting of the power spectrum, $f_{loud}(\boldsymbol{x})$, while F0 is extracted using CREPE [34], denoted by $f_{crepe}(\boldsymbol{x})$, similarly to [2]. We experimented with different representations of F0, such as: octave, note, etc., and achieved similar performance.

In preliminary experiments, we observed that using F0 as input produces inconsistent shakes in the pitch of the generated samples. Therefore, we turn into conditioning on a synthesized melody generated from the F0 instead. The melody is synthesized via a single sinusoid sine-excitation, denoted by $\Gamma(f_{crepe}(\boldsymbol{x}))$.

For speech features, we follow [1] and utilize an intermediate representation from a pre-trained acoustic model optimized for the task of Automatic Speech Recognition (ASR) as an additional input to the model. Specifically, we use the public implementation [35] of Wav2Letter [36], denoted by $f_{w2l}(\boldsymbol{x})$. Since an ASR network is speaker-agnostic by design [37], our method does not require any disentanglement terms or domain-specific (speaker) confusion terms.

Finally, we concatenate and upsample the features in the temporal domain to match the audio frequency.

Single Singer Objective Function We follow the least-squares GAN [38] setup where the discriminator and generator would like to minimize the following terms,

$$\begin{aligned} L_D(D,G,S) &= \sum_{\boldsymbol{x}\in S} [||1 - D(\boldsymbol{x})||_2^2 + ||D(\hat{\boldsymbol{x}})||_2^2] \\ L_{adv}(D,G,S) &= \sum_{\boldsymbol{x}\in S} ||1 - D(\hat{\boldsymbol{x}})||_2^2 \end{aligned} \tag{1}$$

accordingly. S is the set of samples, $\hat{\boldsymbol{x}} = G(z, E(\boldsymbol{x}))$ is the audio sample synthesized from a random noise vector sampled from a uniform distribution $\boldsymbol{z} \sim U(0,1)$, and the concatenated features $E(\boldsymbol{x}) = [f_{loud}(\boldsymbol{x}), f_{w2l}(\boldsymbol{x}), \Gamma(f_{crepe}(\boldsymbol{x}))]$.

In addition, we include a reconstruction loss to further improve optimization stability. We note that two audio samples might be perceptually similar while being the exact opposite in the waveform representation, e.g., in the case of a simple phase inversion (multiply by -1). To mitigate this, we include a spectral amplitude distance loss [12, 39] in multiple FFT resolutions [3, 13, 14]. The spectral amplitude distance loss, for a given FFT size m, is defined as follows:

$$L_{\text{recon}}^{(m)}(G,S) = \sum_{\boldsymbol{x}\in S} \left[\frac{\|\mathcal{S} - \hat{\mathcal{S}}\|_F}{\|\mathcal{S}\|_F} + \frac{\|\log \mathcal{S} - \log \hat{\mathcal{S}}\|_1}{N} \right] \tag{2}$$

where $\|\cdot\|_F$ and $\|\cdot\|_1$ denotes the Forbenius and the L_1 norms, $\mathcal{S} = |\text{STFT}(x)|$ and $\hat{\mathcal{S}} = |\text{STFT}(\hat{x})|$ denotes the Short-time Fourier transform magnitudes of the original and synthesized samples respectively, and N the number of elements. The first term of the expression emphasizes spectral peaks, while the second penalizes silent sections of the audio. The multi-resolution loss is defined as the sum of the above loss for multiple scales:

$$L_{recon}(G,S) = \frac{1}{|M|} \sum_{m\in M} L_{recon}^{(m)}(G,S) \tag{3}$$

where $M = [2048, 1024, 512, 256, 128, 64]$.

Lastly, to further improve the generation quality, we add perceptual losses [40] on top of the generator output. Specifically, we compute the l_1-distance between intermediate activations of the ASR network, h_{w2l} and the CREPE network, h_{crepe} as follows:

$$\begin{aligned} L_{crepe}(G,S) &= \sum_{\boldsymbol{x}\in S} \|h_{crepe}(\boldsymbol{x}) - h_{crepe}(\hat{\boldsymbol{x}})\|_1 \\ L_{w2l}(G,S) &= \sum_{\boldsymbol{x}\in S} \|h_{w2l}(\boldsymbol{x}) - h_{w2l}(\hat{\boldsymbol{x}})\|_1 \end{aligned} \tag{4}$$

Overall, the optimization loss for the generator, G, is defined as:

$$\begin{aligned} L_G(G,D,S) =& L_{recon}(G,S) + \alpha L_{adv}(G,D,S) \\ &+ \beta L_{crepe}(G,S) + \gamma L_{w2l}(G,S) \end{aligned} \tag{5}$$

where α, β, γ are weight factors to balance the contribution of each loss term.

Multi-singer Training Losses In the multi-singer regime, we include a speaker embedding $\boldsymbol{v}_i$ as an additional input to G. The speaker embeddings are learned during training and stored in a Look Up Table. Then, the reconstruction of a sample, $\boldsymbol{x}_i$, pronounced by speaker i is updated to be $\hat{\boldsymbol{x}}_i^i = G(\boldsymbol{z}, E(\boldsymbol{x}_i), \boldsymbol{v}_i)$.

Moreover, we introduce two additional training schemes. The first one is performed by converting a sample from singer i to singer j, while omitting the reconstruction loss. The second one introduces novel virtual training samples by creating parallel samples using back-translation [41] and mixup [42]. This was previously shown by [27] to improve singing voice conversion.

These additional objective functions for unaligned samples are defined as follows,

$$\begin{aligned} L_D^{unaligned}(D,G,S) &= \sum_{\boldsymbol{x}_i\in S} \left[\|1 - D(\boldsymbol{x}_i)\|_2^2 + \|D(\hat{\boldsymbol{x}}_j^i)\|_2^2 \right] \\ L_G^{unaligned}(G,D,S) &= \alpha L_{adv}(G,D,S) + \beta L_{crepe}(G,S) \\ &\quad + \gamma L_{w2l}(G,S) \end{aligned} \tag{6}$$

where $\hat{\boldsymbol{x}}_j^i = G(\boldsymbol{z}, E(\boldsymbol{x}_i), \boldsymbol{v}_j)$. Note the reconstruction loss is omitted, since we do not have the target sample of singer j singing sample $\boldsymbol{x}_i \in S$.

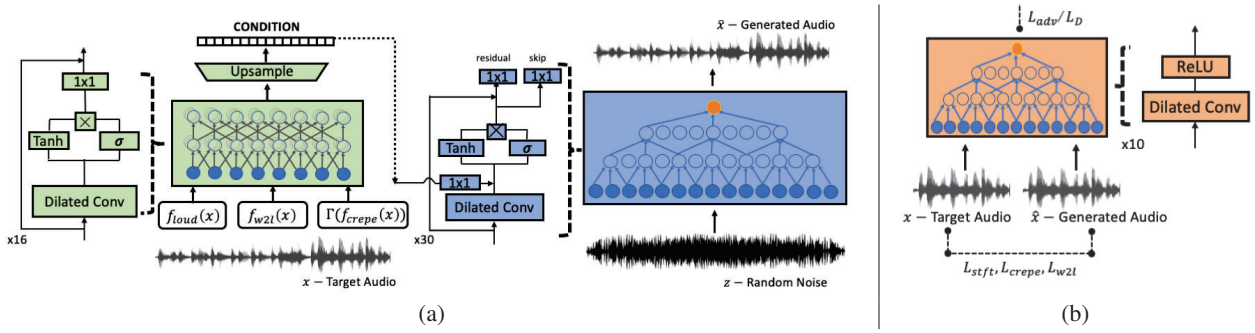

Figure 1: ***Proposed GAN architecture.*** *(a) Generator architecture. Musical and speech features are extracted from a singing waveform $(f_{loud}(x), f_{w2l}(x), \Gamma(f_{crepe}(x)))$ and passed through context stacks (colored green). The features are then concatenated and temporally upsampled to match the audio frequency. The joint embedding is used to condition a non-causal WaveNet (colored blue), which receives random noise as input. (b) Discriminator architecture. Losses are drawn with dashed lines, input/output with solid lines. The discriminator (colored orange) differentiates between synthesized and real singing. Multi-scale spectral loss and perceptual losses are computed between matching real and generated samples.*

To virtually simulate unseen singers we follow the mixup scheme while generating a conversion to a new virtual singer. Specifically, we use a convex combination of two different singers embeddings, $\boldsymbol{v}_j$ and $\boldsymbol{v}'_j$, as:

$$\boldsymbol{u} = \nu \boldsymbol{v}_j + (1-\nu)\boldsymbol{v}_{j'} \tag{7}$$

where $\nu \sim U[0,1]$ is drawn from the uniform distribution. Sample $\boldsymbol{x}^j_{\boldsymbol{u}} = G(\boldsymbol{z}, E(\boldsymbol{x}_j), \boldsymbol{u})$ is then generated from $\boldsymbol{x}_j$, and translated back to singer j as follows: $\hat{\boldsymbol{x}}^j_{\boldsymbol{u}} = G(\boldsymbol{z}, E(\boldsymbol{x}^j_{\boldsymbol{u}}), \boldsymbol{v}_j)$. The produced set of artificial examples is denoted as $S_{mixup} = \{(\boldsymbol{x}_j, \hat{\boldsymbol{x}}^j_{\boldsymbol{u}})\}$. Finally, the discriminator and generator are optimized by minimizing the loss over supervised, unaligned and virtual samples:

$$\begin{aligned} L_D^{multi}(G,D) =& L_D(G,D,S) + L_D^{unaligned}(G,D,S) \\ &+ L_D(G,D,S_{mixup}) \\ L_G^{multi}(G,D) =& L_G(G,D,S) + L_G^{unaligned}(G,D,S) \\ &+ L_G(G,D,S_{mixup}) \end{aligned} \tag{8}$$

Note that in the mixup setting, the $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}^j_{\boldsymbol{u}}$.

Architecture Generator G, is based on a non-causal WaveNet architecture [3, 43]. It receives as input a random noise vector sampled from a uniform distribution, $\boldsymbol{z} \sim U(0,1)$ and the input features described above. Each input feature (f_{loud}, f_{w2l} and f_{crepe}) is passed through a separate convolutional stack [2, 32], which is composed of two blocks of eight non-causal convolutional layers. The layers in each block have an exponentially increasing dilation rate. Each layer employs 128 filters and a kernel-size of 3. The features are then upsampled by a series of interleaved nearest neighbor upsampling and convolutional layers. Once temporally-aligned to the audio, the features are concatenated to form the conditioning signal.

The generator is composed of a series of 30 non-causal layers ordered in three blocks. The dilation rate in a single block of layers is exponentially increasing. Thus, the model has a receptive field of 3,072 samples, which means each sample is generated based on a window of $96ms$ in future and past directions. Each layer has 128 residual-channels, 128 skip-channels and a kernel-size of 3. Our model achieves inference speed of 10.14 times faster than real-time, on a single Tesla V100 GPU.

The discriminator is composed of ten layers of 1-D convolutions followed by a leaky-ReLU activation with a leakiness of 0.2, with a linearly increasing dilation rate. Each convolution layer consists of 128 filters with a kernel-size of 3. The discriminator outputs a prediction for each time-step in the input audio signal. The loss is computed by averaging across the time-domain. Both the discriminator and generator apply weight normalization [44] on all layers.

4. Experiments

We perform a series of experiments to evaluate the proposed method against several baselines. We experimented with generating singing using speech-only, singing-only and mixed datasets. We explore both many-to-one conversion on a large single identity corpus and many-to-many conversion using learned identities from a corpus with a variable amount of audio per identity. Moreover, we perform an extensive ablation study to better understand the contribution of each component. Audio samples are available online at `https://singing-conversion.github.io/`, as well as in the supplementary material.

Datasets We report results on several datasets. *LJ* [45] is a large single speaker speech corpus with approximately 24 hours of audio recording. *LCSING* is a studio recordings of a single professional singer [46], which was filtered using an off-the-shelf voice activity detector and contains 3 hours and 40 minutes of very expressive and high dynamic range recordings, including some melodic singing without lyrics. For the multi-speaker experiments, we use the speech corpus, *VCTK* [47], to learn 109 singers with 44 hours of audio recordings. Finally, the *NUS-48E* [48] dataset, which includes six male singers and six female singers, has both singing and reading of four songs per voice, resulting in a total of 15 minutes per singer. All audio was down-sampled to 16kHz with a single channel. All datasets were randomly split according to a 80%/10%/10% of train/val/test partitions.

Hyperparameters We train our models for 800K steps, with a batch-size of 8 one second long audio segments. We use RADAM [49] optimizer with a learning rate of 0.0001. The learning rate is halved every 200K steps. The discriminator joins the training process after 100K steps and the perceptual losses after 50K steps. For the CREPE perceptual loss, we use the intermediate-activation before the final sigmoid activation. For the ASR network loss, we use the output of the tenth convolutional block. We use $\alpha = 4, \beta = 1, \gamma = 10$ for the weight factors in Eq. 5. The model is trained with a mixup batch every 3 steps after 100K steps of training had passed.

Table 1: *Test scores for singing voice conversion. For MOS, SIM, identification, higher is better. For VDE and FFE – lower.*

Dataset	Method	MOS ↑	SIM ↑	Identification ↑	VDE ↓	FFE ↓
Source Singing	Ground Truth	4.10±0.84	86.11%	100%	—	—
LJ	Mellotron	3.79±1.06	59.04%	76.47%	8.35%	9.50%
	Ours	4.06±0.81	60.46%	97.87%	4.19%	5.51%
	Ground Truth	4.51±0.70	70.83%	97.91%	—	—
VCTK	Mellotron	3.14± 0.82	70.00%	57.67%	15.02%	16.62%
	Ours	3.88± 0.56	78.57%	96.41%	6.98%	7.72%
	Ground Truth	4.35± 0.74	75.00%	99.28%	—	—
LCSING	USVC	3.52±0.91	38.24%	8.51%	8.19%	11.46%
	Ours	3.81±0.92	67.74%	100%	4.00%	4.80%
	Ground Truth	3.95±0.73	77.78%	100%	—	—
NUS-48E	Mellotron	3.55±0.87	69.44%	60.11%	6.59%	8.02%
	WGANSing	3.60±0.94	80.56%	92.27%	3.85%	5.06%
	USVC	3.78±0.85	61.90%	93.45%	4.82%	20.80%
	Ours	4.04±0.68	81.82%	97.02%	2.47%	3.50%

Singing conversion All experiments were performed by converting singing recordings of the NUS-48E to the target identities learned from the datasets described above. Therefore, experiments conducted on the LJ, LCSING and VCTK evaluate the methods' invariance to the *input* voice identity, while experiments conducted on the NUS-48E dataset, evaluate the methods' performance while converting across a fixed set of singers.

Evaluation metrics are based on subjective and objective success metrics: (i) Mean Opinion Scores (MOS), human raters rate the naturalness of the audio samples on a scale of 1–5. Each experiment, included 40 randomly selected samples rated by 20 raters. (ii) ABX testing for similarity, in which we present each rater with two audio samples A and B. The examples originate from two different singers. These two samples are followed by a third utterance X randomly selected to be from the same identity as A or B. Next, the rater must decide whether X has the same identity as A or B. We report the success rate across all raters. (iii) Automatic identification metric by training a multi-class classifier on the ground-truth training partitions of all datasets and reporting the success rate of the classifier, similar to [50–52]. (iv) Voicing Decision Error (VDE) [53], which measures the portion of frames with voicing decision error, (v) F0 Frame Error (FFE) [54], measures the percentage of frames that contain a deviation of more than 20% in pitch value or have a voicing decision error.

Mellotron [21] showed good results on singing generation by training on speech datasets. Therefore, we use it as the baseline for the experiments on speech datasets. Given a template singing sample, Mellotron extracts the rhythm, the alignment between text and spectral features, which is then used to generate the conversion. For emotive samples, like in the NUS-48E dataset, the rhythm extraction sometime failed. Therefore, we used a forced-aligner [55] to create a synthetic alignment map and replace the rhythm extracted by Mellotron with the synthetic one before generating the conversion.

For a fair comparison we do not report Mellotron results on LCSING dataset due to instability in the training caused by the following reasons. In the LCSING dataset there are only 40 minutes of transcribed segments and many of the recordings contain non-lexical vocables.

Our experiments on NUS-48E dataset involved baselines which showed convincing results: WGANSing [23] and Unsupervised Singing Voice Conversion (USVC) [27]. Both methods require multi-singer voice dataset for training. Therefore, we do not apply them on LJ and VCTK. For the single singer dataset, we use the same architecture as USVC but without the confusion term.

Table 2: *Ablation study on the LJ dataset.*

Method	MOS ↑	VDE ↓	FFE ↓
F0 condition	2.37±0.93	7.20%	8.81%
Melody condition	3.26±1.04	6.33%	7.90%
+ ASR perceptual loss	3.36±0.96	6.47%	8.03%
+ CREPE perceptual loss	4.06±0.82	4.19%	5.51%

Table 1 presents the results for all of the above models. On speech datasets, LJ and VCTK, our models outperform Mellotron, despite the latter utilizing the underlying text as input. In addition to subjective quality, our method predicts both voicing decision and pitch accuracy better than Mellotron. Results on LCSING, show that our method is better than the baseline and is able to generate recognizable samples. On a multi-singer dataset, NUS-48E, our method generates subjectively higher quality samples, which are more identifiable than the baselines.

Ablation We perform ablation for the suggested single singer training framework. Table 2 summarizes the results. The comparison between the F0 condition and the Melody condition, shows that providing melody as input to the model reduces the overall error with regard to pitch generation. The addition of the ASR perceptual loss slightly improves the model MOS scores at the cost of slightly reducing the pitch metrics. Adding the CREPE perceptual loss adds a significant gain to the model performance across all metrics.

5. Conclusion

We present an unsupervised method that can convert a singing voice to a voice that is sampled either as speaking or singing. The method employs multiple pre-trained encoders and perceptual losses and achieves state of the art results on both objective and subjective measures. Conditioning the generator on a sine-excitation was shown to be beneficial while further improving the results. As future work, we would like to focus on temporal modification of the input singing to further match the style of the target singer.

6. References

[1] A. Polyak, L. Wolf, and Y. Taigman, "TTS Skins: Speaker Conversion via ASR," *arXiv:1904.08983*, 2019.

[2] L. Hantrakul, J. Engel, A. Roberts, and C. Gu, "Fast and flexible neural audio synthesis," in *ISMIR*, 2019.

[3] 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 *ICASSP*, 2020.

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

[5] S. Mehri *et al.*, "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model," in *ICLR*, 2017.

[6] N. Kalchbrenner *et al.*, "Efficient Neural Audio Synthesis," in *ICML*, 2018.

[7] A. van den Oord *et al.*, "Parallel WaveNet: Fast high-fidelity speech synthesis," in *ICML*, 2018.

[8] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," *ICLR*, 2019.

[9] S. Kim, S.-G. Lee, J. Song, J. Kim, and S. Yoon, "FloWaveNet : A generative flow for raw audio," in *ICML*, 2019.

[10] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP*, 2019.

[11] K. Kumar *et al.*, "MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis," in *NIPS*, 2019.

[12] S. Ö. Arık, H. Jun, and G. Diamos, "Fast spectrogram inversion using multi-head convolutional neural networks," in *IEEE Signal Processing Letters*, 2018.

[13] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," *ICLR*, 2020.

[14] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter-based waveform model for statistical parametric speech synthesis," in *ICASSP*, 2019.

[15] J. Bonada, M. Umbert, and M. Blaauw, "Expressive singing synthesis based on unit selection for the singing synthesis challenge 2016," in *INTERSPEECH*, 2016.

[16] K. Saino, H. Zen, Y. Nankaku, A. Lee, and K. Tokuda, "An hmm-based singing voice synthesis system," in *Ninth International Conference on Spoken Language Processing*, 2006.

[17] K. Oura *et al.*, "Recent development of the hmm-based singing voice synthesis system—sinsy," in *Seventh ISCA Workshop on Speech Synthesis*, 2010.

[18] K. Nakamura *et al.*, "HMM-based singing voice synthesis and its application to japanese and english," in *ICASSP*, 2014.

[19] M. Blaauw and J. Bonada, "A Neural Parametric Singing Synthesizer," in *INTERSPEECH*, 2017.

[20] M. Blaauw, J. Bonada, and R. Daido, "Data efficient voice cloning for neural singing synthesis," in *ICASSP*, 2019.

[21] R. Valle, J. Li, R. Prenger, and B. Catanzaro, "Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens," *ICASSP*, 2020.

[22] M. Blaauw and J. Bonada, "Sequence-to-Sequence Singing Synthesis Using the Feed-Forward Transformer," in *ICASSP*, 2020.

[23] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "WGANSing: A multi-voice singing voice synthesizer based on the wasserstein-gan," in *European Signal Processing Conference*, 2019.

[24] K. Kobayashi *et al.*, "Statistical singing voice conversion based on direct waveform modification with global variance," in *INTERSPEECH*, 2015.

[25] ——, "Statistical singing voice conversion with direct waveform modification based on the spectrum differential," in *INTERSPEECH*, 2014.

[26] F. Villavicencio and J. Bonada, "Applying voice conversion to concatenative singing-voice synthesis." in *INTERSPEECH*, 2010.

[27] E. Nachmani and L. Wolf, "Unsupervised Singing Voice Conversion," *INTERSPEECH*, 2019.

[28] C. Deng *et al.*, "Pitchnet: Unsupervised Singing Voice Conversion with Pitch Adversarial Network," in *ICASSP*, 2020.

[29] Y.-J. Luo *et al.*, "Singing Voice Conversion with Disentangled Representations of Singer and Vocal Technique Using Variational autoencoders," in *ICASSP*, 2020.

[30] X. Chen, W. Chu, J. Guo, and N. Xu, "Singing voice conversion with non-parallel data," *arXiv:1903.04124*, 2019.

[31] G. Xiaoxue *et al.*, "Personalized singing voice generation using wavernn," in *Odyssey*, 2020.

[32] C. Hawthorne *et al.*, "Enabling factorized piano music modeling and generation with the maestro dataset," in *ICLR*, 2019.

[33] B. C. Moore, B. R. Glasberg, and T. Baer, "A model for the prediction of thresholds, loudness, and partial loudness," *Journal of the Audio Engineering Society*, vol. 45, no. 4, pp. 224–240, 1997.

[34] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *ICASSP*, 2018.

[35] J. Li *et al.*, "Jasper: An end-to-end convolutional neural acoustic model," *INTERSPEECH*, 2019.

[36] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2Letter: an End-to-End Convnet-based Speech Recognition System," *CoRR*, vol. abs/1609.03193, 2016.

[37] Y. Adi *et al.*, "To reverse the gradient or not: An empirical comparison of adversarial and multi-task learning in speech recognition," in *ICASSP*, 2019.

[38] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *ICCV*, 2017.

[39] A. v. d. Oord *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," *ICML*, 2018.

[40] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.

[41] R. Sennrich *et al.*, "Improving Neural Machine Translation Models with Monolingual Data," in *ACL*, 2016.

[42] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR*, 2018.

[43] D. Rethage, J. Pons, and X. Serra, "A wavenet for speech denoising," in *ICASSP*, 2018.

[44] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *NIPS*, 2016.

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

[46] L. Crew, http://lucillecrew.com//, 2019.

[47] C. Veaux *et al.*, "CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit," 2017.

[48] Z. Duan *et al.*, "The nus sung and spoken lyrics corpus: A quantitative comparison of singing and speech," in *Asia-Pacific Signal and Information Processing Association Conference*, 2013.

[49] L. Liu *et al.*, "On the variance of the adaptive learning rate and beyond," in *ICLR*, 2020.

[50] S. Arik *et al.*, "Deep voice 2: Multi-speaker neural text-to-speech," in *NIPS*, 2017.

[51] Y. Taigman, L. Wolf, A. Polyak, and E. Nachmani, "VoiceLoop: Voice Fitting and Synthesis via a Phonological Loop," in *ICLR*, 2018.

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

[53] T. Nakatani *et al.*, "A method for fundamental frequency estimation and voicing decision: Application to infant utterances recorded in real acoustical environments," *Speech Communication*, 2008.

[54] W. Chu and A. Alwan, "Reducing f0 frame error of f0 tracking algorithms under noisy conditions with an unvoiced/voiced classification frontend," in *ICASSP*, 2009.

[55] M. McAuliffe *et al.*, "Montreal forced aligner: Trainable text-speech alignment using kaldi." in *INTERPSEECH*, 2017.