TTS Skins: Speaker Conversion via ASR

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

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

adampolyak@fb.com

Abstract

We present a fully convolutional wav-to-wav network for converting between speakers' voices, without relying on text. Our network is based on an encoder-decoder architecture, where the encoder is pre-trained for the task of Automatic Speech Recognition, and a multi-speaker waveform decoder is trained to reconstruct the original signal in an autoregressive manner. We train the network on narrated audiobooks, and demonstrate multi-voice TTS in those voices, by converting the voice of a TTS robot.

Index Terms: voice conversion, text to speech, human-computer interaction.

1. Introduction

Text-To-Speech (TTS) and Automatic Speech Recognition (ASR) are key components in human-machine interfaces, deployed on popular virtual assistants. Recent advances in neural networks have dramatically increased the naturalness of TTS systems in synthesizing artificial human speech from text, thereby increasing their accessibility to users.

These systems are trained end-to-end, from text to speech, on about 100 hours of a single speaker, usually recorded in a studio. Enriching this interface by providing additional speakers can either be done by collecting a similar dataset for each speaker, or through some adaptation process. Currently, most adaptation methods have shown limited success, either by requiring a similar data collection effort or by providing inferior results with respect to the identity of the speaker.

We present a fully convolutional encoder-decoder network, that can be plugged into existing TTS systems and provide additional speaker identities ("skins"), requiring a considerably smaller quantity of target speaker recordings and without text. Crucially, our encoder utilizes an ASR network that provides both high-level acoustic perception, as well as speaker-independent features.

Our main contributions are: (i) We present the utility of pretrained ASR networks as suitable encoders for a voice conversion network. (ii) We present a conditioned WaveNet decoder based on these features. (iii) We propose a multispeaker TTS solution that has favorable properties, with regards to the required training data for the majority of the voices.

2. Related Work

In addition to factors such as the expected quality and the computational complexity, a major factor in the applicability of a specific voice conversion technique is the type of supervision the method requires. Specifically, the quantity of the data that fitting a target speaker requires is of a great importance. Unlike many recent methods, our method does not require parallel training data between speakers and the reference source voice need not be available during training. The amount of data required varies between the phases of training. Since a high capacity WaveNet decoder is used, considerable (unlabeled) data is required for the training voices used for the initial training phase, in which the network and the look up table are being trained. Then, when fitting additional voices, much less data is required.

The method of [1] obtains data efficiency, by performing the conversion on a shared intermediate vocoder representation. This intermediate representation is estimated from linguistic features and decoded with a WaveNet. The WaveNet decoder is generic (from the vocoder features to waveform). However, during fitting, both the part that maps the linguistic features to the vocoder features and the WaveNet are fitted to each new speaker. In contrast to our system, which uses an off-the-shelf ASR network, the linguistic features in [1] are trained (on a proprietary dataset) together with the subsequent networks.

The topic of data efficiency was recently also studied in the supervised TTS case, where the training set contains both audio and text [2]. Their conclusion, similarly to ours, is that it is beneficial to both optimize the embedding vector of the new speaker, as well as adopt the weights of the network itself.

Many of the literature voice conversion methods that require parallel data between the speakers, which is a limiting factor by itself, also require the parallel samples to be aligned in time. This requirement can be overcome in an iterative fashion, by first performing a nearest neighbor search of parallels and then refining the conversion module. Erro et al. [3] have shown that repeating the matching process with the gradually improving conversion module leads to better and better results. A similar iterative process was used by Song et al. [4], in which the transformation takes the form of adapting GMM coefficients. ASR features were used by Xie et al. [5] in order to align multiple speakers in the latent space of the ASR system. Such an alignment relies on the ability of such features to be (mostly) speaker-agnostic. Once the alignment was formed, it was used in order to convert between voices, by concatenating matching fragments, i.e., in a unit-based approach.

In this work, we rely on a trained ASR network in order to obtain features. The ASR features are supposedly mostly orthogonal to the identity. An alternative approach would be to use the speaker identification features, in order to represent the source and target voices and perform the transformation between the two. I-vectors [6] are a GMM-based representation, which is often used in speaker identification or verification. Kinnunen et al. [7] have aligned the source and target GMMs, by comparing the i-vectors of the two speakers, without using transcription or parallel data. Unlike our method, the reference speaker is known at training time, and their method employs an MFCC vocoder, which limits the output's quality, in comparison to WaveNets. Speaker verification features were also used to embed speakers

---

by Jia et al. [8]. In this case, the speaker embedding is based on a neural classifier.

While in our method, the encoder is pre-trained for the purpose of ASR, other voice conversion methods employ a neural autoencoder. Specifically, many methods are based on variational auto encoders [9]. Hsu et al. [10], employs, similarly to us, one encoder and a parameterized decoder, where the parameters represent the identity. In a follow-up work [11], a WGAN [12] loss term was added, in order to improve the naturalness of the resulting audio. In these variational autoencoder works, spectral frames are used, while our WaveNet decoder uses the waveform.

A recent line of work [13, 14, 15, 16] employs an ASR encoder to extract Phonetic Posterior Grams (PPGs), to perform voice conversion without parallel data. PPGs is a vector of probabilities for each phonetic class for a specific time-frame [17]. Sun et al. [13] predict STRAIGHT vocoder features from PPG features, assumed to be speaker independent, allowing them to perform many-to-one conversion. Saito et al. [14] present a method based on VAE for non-parallel voice conversion, conditioned on both speaker representations and phonetic content. Zhao et al. [15, 16] used PPG to perform accent conversion - first using conventional vocoders following with neural vocoders. Our work differs by (i) using non-interpretable features rather than PPGs, and (ii) predicting wavform directly from those features rather than relying on spectral features (mel-spectrogram or vocoder features). Tian et al. [18] presented a vocoder-free method for speaker conversion. Like us, they do not rely on intermediate spectral representation and directly feed phonetic context represented as PPGs to a WaveNet neural vocoder. Experimentally, we compare two different PPG representations to our method and show that our method leads to more appealing audio samples.

An audio conversion method based on a waveform WaveNet autoencoder was presented in [19]. Trained mostly to convert between musical domains, voice conversion results are also presented. Unlike our method, the encoder is learned and multiple decoders are used, one per speaker. A single decoder that, similarly to ours, is conditioned on the identity of the target speaker was used in [20]. While, we employ a look up table that stores continuous speaker embeddings, their work employs a one-hot encoding. More importantly, the approach for obtaining a speaker-agnostic representation, is different. In [20], the capacity of the latent space was reduced, by employing a discrete representation, while in [19], a domain confusion loss is used. In our work, we employ neither approach, and instead rely on the speaker-agnostic nature of the ASR features.

3. Method

The voice conversion method is based on training an autoencoder architecture, conditioned on an embedding of one of the k speakers. In our method, the encoder is taken to be a subnetwork of a pre-trained ASR network, specifically the public implementation [21] of Wav2Letter [22]. Since an ASR network is used, by design, it is invariant to the speaker [23] and the method does not require any disentanglement terms or other domain (speaker) confusion terms. This greatly adds to the stability of the method. During training, the speaker embeddings are learned and a softmax reconstruction loss is applied to the output of the shared WaveNet decoder.

A diagram of the architecture is shown in Fig. 1. It includes the pre-trained encoder E, the learned speaker embeddings and a single decoder, which is shared among all speakers, D.

The autoregressive decoder D receives as input the sequence that was generated so far and is conditioned on: (i) the latent representation produced by the encoder E, and (ii) the desired speaker embedding. The first conditioning signal is obtained by upsampling this encoding in the temporal domain to obtain the original audio rate. Upsampling is done by repeating each conditioning frame to match the audio frequency. It is then applied, after the 128-dimensional speaker embedding is concatenated to it, to each of the WaveNet layers.

Training and the Losses Used. Since the encoder is taken off-the-shelf, we only train the decoder D. Let $D(\cdot, v)$ denote the conditioning of the decoder on a speaker that is embedded in a vector v. During training, we fit the vector v_j of each speaker j and store these in a Look Up Table (LUT). The following optimization problem is defined:

$$\min_{\theta_D, v_1, \ldots, v_k} \sum_{j=1}^{k} \sum_{s^j} \mathcal{L}(D(E(s^j), v_j), s^j), \qquad (1)$$

where θ_D are the parameters of the decoder, s_j are the training samples of speaker j, k is the number of speakers and $\mathcal{L}(o, y)$ is the cross entropy loss, applied to each element of the output o and the corresponding element of the target y separately. Note that the decoder D is an autoregressive model that is conditioned on the output of E in the previous frames. During training we use "teacher forcing", the autoregressive model is fed the target output s^j from the previous time-step, instead of the generated output.

Once the network is trained, we can convert a new voice sample s (of an existing or a new speaker) to an existing speaker j. This is done by applying the autoencoder pathway of speaker j, in order to obtain a conversion to this voice: $D(E(s), v_j)$.

Following Chen et al. [2], we present a variant of our method, which adds the fundamental frequency values $F0$ as additional conditioning to the network. $F0$ values were extracted at 5 ms steps using WORLD [24] vocoder. In the unvoiced regions, the values were set to a constant value across all speakers. This enables the model to better preserve the prosody of the original voice sample s. Following this variant, the optimization problem is now defined as:

$$\min_{\theta_D, v_1, \ldots, v_k} \sum_{j=1}^{k} \sum_{s^j} \mathcal{L}(D(E(s^j), v_j, F0(s^j)), s^j), \quad (2)$$

where $F0(s^j)$ is the fundamental frequency series extracted from input voice sample s. Similarly, voice conversion for speaker j is obtained via $D(E(s), F0(s), v_j)$.

Fitting New Speakers. It is desired to be able to add additional voices to the network post training, preferably with little training data and using a shorter training procedure. In order to achieve this, we perform fine-tuning of the pre-trained network, using the samples s^{k+1} of the new speaker. In such a case, we fit both the embedding v_{k+1} and fine tune the weights of the decoder D. Furthermore, instead of using randomly initialized embeddings for the new speakers, as was done in [8], the initial embedding of the new speaker v_{k+1}^0 is initialized with the best available embedding out of the training speakers. This is done by finding the speaker embedding which minimizes the loss in Eq. 2 for samples s^{k+1} of the new speaker:

$$v_{k+1}^0 = \arg\min_{v_1, \ldots, v_k} \mathcal{L}(D(E(s^{k+1}), v_j, F0(s^{k+1})), s^{k+1}). \quad (3)$$

After initialization, the loss of Eq. 2 is used to refine the embedding v_{k+1} and the decoder D, without reintroducing the training samples of the other speakers.
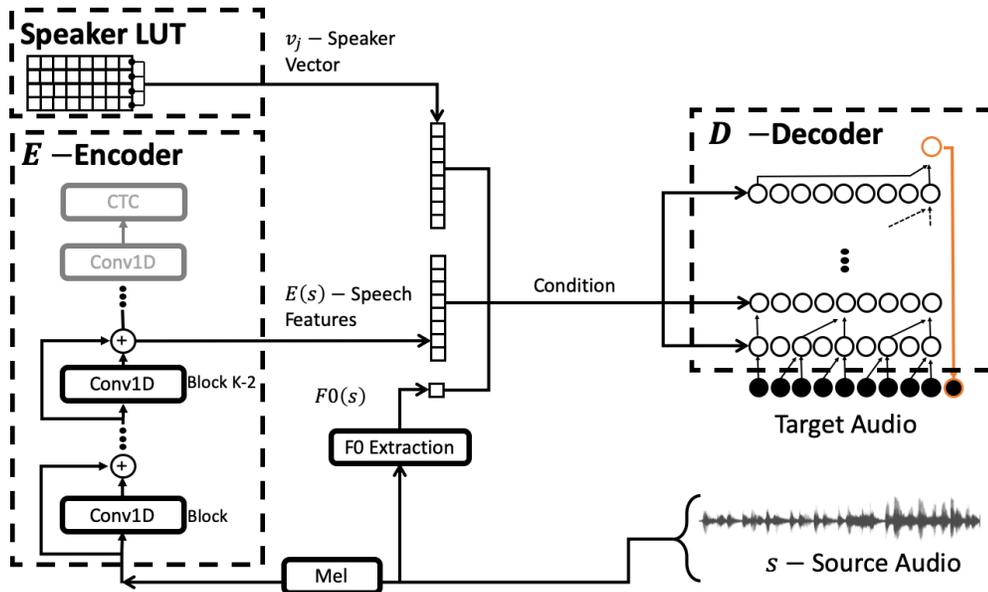
Figure 1: ***The architecture of our method.*** *The non-causal pre-trained encoder embeds the source audio signal, s, to speech features, $E(s)$. Then, the speaker's embedding, v_i, and the fundamental frequency values, $F0(s)$, are concatenated to the speech features. The joint embedding is used to condition the autoregressive decoder, D, at multiple layers. During training, the decoder's input is the source audio, instead of the generated one.*

Network Architecture. The encoder E is a deep time delay neural network (TDNN), comprised of blocks of 1D-convolutional layers. It operates on log-mel filterbank extracted from a 16 kHz speech signal, using a sliding window of 20 ms with hops of 10 ms. Initially, the mel features sequence is down-sampled by $\times 2$, using a single strided convolutional layer. The downsampeld input then passes a series of 10 1D-convolutional blocks, each with 5 convolutional layers. The 1D-convolutional layers are composed of a sequence of a convolutional operation, batch normalization, clipped ReLU activation, and dropout. Blocks are composed of a series of the former layer, with a residual connection adding a projection of the block input to the block output. The model was trained on Librispeech [25] together with data augmentation. We use the activations of layer `conv105` as input to the decoder, resulting a 768-dimensional conditioning signal at 50 Hz.

The WaveNet decoder D has four blocks of ten dilated residual convolution layers [26]. The layers in each block are with exponentially increasing dilation rate. We use 128 residual channels and 128 skip channels across all of the layers and a kernel-size of 2. At the top of the WaveNet, are two fully connected layers and a softmax activation, which provides probabilities for the quantized audio output (256 options) at the next timeframe. The autoregressive decoder generates audio at 24 kHz sampling frequency, resulting receptive field of 125ms (4,093 samples). We use 24 kHz sampling rate as it was shown to achieve higher subjective ratings [27]. In each WaveNet layer, the conditioning signal is passed to through a 1×1 convolutional layer, that is unique for each conditioning location.

We have developed efficient CUDA kernels that can decode autoregresively in real time (less than one second processing time for one second of output), thereby enabling efficient conversion. These kernels are loosely based on the nv-wavenet kernels (https://github.com/NVIDIA/nv-WaveNet) provided by NVIDIA, and were optimized to utilize NVIDIA's Volta architecture. Additional modifications are based on the WaveNet variant of [28] : (i) we add terms to ResNet skip connections that are directly computed from the WAV, and (ii) we add conditioning to the last fully connected layer.

4. Experiments

We present a series of experiments evaluating our method on: (i) many-to-many natural speech conversion on two public datasets, LibriTTS [27] and VCTK [30] from speakers seen during training, (ii) many-to-many natural speech conversion on the public datasets from speakers unseen during training, (iii) the customization of an existing off-the-shelf state of the art TTS engine [31] with identities from the public datasets and (iv) learning new identities with limited amount of data (less than five minutes) on the voice conversion challenge 2018 [32]. Our samples can be found at https://tts-skins.github.io, as well as in the supplementary.

LibriTTS is derived from the LibriSpeech corpus [25] by removing noisy recordings, splitting sentences and 24kHz sample rate. We use speakers with at least five minutes of recordings from the 'train-clean-100' partition, as a result 199 speakers remain. In addition, we use the 'test-clean' partition, which includes 39 additional speakers, to evaluate our method on conversion of speakers unseen during training. The VCTK dataset is downsampled to 24kHz and eleven speakers are held-out for unseen speakers evaluation. Both datasets are split to 80% train, 10% validation and the rest is used for testing. On each dataset, we train a single ASR-features-conditioned WaveNet decoder.

Speaker conversion experiments. We use both user-study and automatic based success metrics: (i) Mean Opinion Scores (MOS), on a scale between 1–5, which were computed using the crowdMOS [33] package. (ii) Mel cepstral distortion (MCD) between synthesized utterances and the reference utterances of the same speaker. Dynamic time warping is used to calculate the distortion for unaligned sequences. (iii) Speaker classification is employed to evaluate the capability of the method to synthesize distinguished voices. Similar to [34, 35, 36], automatic speaker identification results are obtained by training a multi-class CNN on the ground-truth training set of multiple speakers, and tested on the generated ones. The network operates on the WORLD vocoder features [24] extracted from the input and employs five batched-normalized convolutional layers of 3×3 filters with 128 ReLU activated channels. This is followed by max-pooling, average pooling over time, two fully-connected layers, and ending with a softmax corresponding to the number of speakers in

Table 1: *Test scores (Mean ± SD) for LibriTTS and VCTK. For MOS and identification, higher is better. For MCD – lower.*

Data	Method	LibriTTS			VCTK		
		MOS	MCD	Identification	MOS	MCD	Identification
Source voice	Ground truth	4.45±0.68	—	98.42	4.39±0.78	—	99.5
Seen during training	Full method	**3.78±0.83**	—	**96.12**	**4.08±0.75**	**8.76±1.72**	**98.97**
	w/o F0	3.61±0.83	—	96.96	3.59±0.96	8.99±1.58	96.89
	Autoencoder baseline	2.89±0.88	—	29.19	3.46±1.07	9.45±1.63	69.26
	PPG	2.82±0.91	—	94.01	2.67±0.93	9.19±1.50	98.77
	PPG2	2.87±1.00	—	95.77	3.03±1.06	9.18±1.52	96.24
Unseen during training	Full method	**3.70±0.80**	—	**97.10**	**4.05±0.74**	**8.94±1.53**	**98.33**
	w/o F0	3.67±0.82	—	97.15	3.62±0.99	9.25±1.62	95.69
	Autoencoder baseline	3.02±0.89	—	32.55	3.83±0.91	9.65±1.51	66.20
	PPG	2.79±0.93	—	94.05	2.89±0.93	9.45±1.45	97.45
	PPG2	2.71±0.93	—	95.43	3.19±1.04	9.79±1.86	97.25
TTS robot	Unconverted	4.25±0.77	10.12±1.27	—	4.37±0.80	14.52±2.40	—
	Full method	**3.67±0.81**	**8.13±0.95**	**96.06**	**4.17±0.88**	**12.68±2.17**	**99.25**
	w/o F0	3.47±0.76	8.43±0.97	96.66	3.75±1.07	13.06±2.26	96.36
	Autoencoder baseline	3.02±0.84	9.38±1.09	60.26	3.85±1.05	13.81±2.29	75.56
	PPG	2.91±0.94	8.52±0.93	96.63	3.50±0.83	12.45±1.92	98.36
	PPG2	2.85±0.87	8.76±1.06	95.08	3.66±1.03	12.57±2.10	97.62

Table 2: *MOS and similarity scores for the 2018 Voice Conversion Challenge (Mean ± SD; higher is better)*

Method	Hub		Spoke	
	MOS	Similarity	MOS	Similarity
GT source	4.36±0.64	1.72±1.04	4.35±0.56	1.68±1.09
GT target	4.36±0.66	3.69±0.63	4.42±0.56	3.72±0.59
N10 [1]	3.92±0.75	2.83±1.20	3.98±0.52	3.13±0.97
N17 [29]	3.27±0.95	2.77±1.17	3.40±0.88	3.05±0.96
Ours	3.84±0.85	2.87±1.14	4.00±0.55	3.14±0.97

the dataset. Tab. 1 depicts the results of our method.

In experiments (i-iii), we compare with three baselines. The first is the WaveNet autoencoder voice conversion system of [19]. The second is a speaker conversion via PPG features marked as 'PPG'. PPG features were extracted using a p-norm DNN [37], as done by [16]. Finally, an additional PPG based speaker conversion system is trained by using a different set of PPG features, extracted by using pytorch-kaldi [38] framework. The underlying ASR model achieves an accuracy of PER=14.9% on the TIMIT dataset [39]. It is marked as 'PPG2'.

As can be seen, our full method improves the metrics over the baselines in all three settings. Furthermore, adding $F0$ improves the quality of our method output. Additionally, on the VCTK dataset, it also improves the identifiability of the generated samples. Finally, the quality of samples generated from voices seen in the training set is preferred over samples generated by converting samples from unseen speakers.

The voice conversion challenge 2018. VCC2018 evaluated 23 voice conversion systems on two tasks. First, the Hub task, in which participants were asked to convert between speakers with parallel training corpora. Second, the Spoke task, for which participants were given training data for source and target speakers that contained different sets of utterances. Each task included four source speakers and four target speakers. Source speakers of the Hub task were all different from the source speakers in the

Spoke task. Both tasks shared the same group of target speakers. The training data for each speaker (either source or target) is composed of 81 samples of recorded speech, the total amount of time varies between 4-5 minutes.

In each task, participants are required to transform 35 utterances from each source speaker to each target speaker. As a result, each task requires the generation of $4 \times 4 \times 35 = 560$ samples. Samples are evaluated subjectively for their quality and similarity to a reference samples recorded by the target speakers. In our experiments, we used MOS for both quality and similarity evaluation. Quality of generated samples was rated on a scale of 1–5. Similarity was rated on a scale of 1–4, representing a binary similarity value (similar or not) and two values for certainty (sure/unsure) which included the amount of confidence. Following the challenge, the organizers published the submitted samples making it a suitable test bed for evaluating our method's ability to adapt to new speakers based on a limited amount of data, as described above in Sec. 3.

Tab. 2 shows ours results versus the best performing method of the VCC 2018, which is known as N10 [1]. In addition, we included the results of N17 [29], which matched the similarity scores of N10 on the Hub task. As can be seen, our system performs directly on par with the best performing system of the VCC 2018.

5. Discussion

This work enables customization of text-to-speech engines to an unlimited number of speakers. We show that combining a neural encoder obtained from an automatic speech recognition network with a neural audio decoder, achieves promising results. The alternative methods struggle to disentangle the speaker identity at the encoder level, while an ASR based encoder is trained to be speaker-agnostic.

In addition to enabling multiple voice persona, the new technology can be readily used to create voice effects, e.g., convert ones voice to a whimsical voice. Being able to generate a large variety of voices is also beneficial to the study of identifying manipulated audio.

6. References

[1] L.-J. Liu *et al.*, "WaveNet vocoder with limited training data for voice conversion," *INTERSPEECH*, 2018.

[2] Y. Chen *et al.*, "Sample efficient adaptive text-to-speech," in *ICLR*, 2019.

[3] D. Erro, A. Moreno, and A. Bonafonte, "INCA algorithm for training voice conversion systems from nonparallel corpora," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 5, 2010.

[4] P. Song, Y. Jin, W. Zheng, and L. Zhao, "Text-independent voice conversion using speaker model alignment method from non-parallel speech," *INTERSPEECH*, 2014.

[5] F.-L. Xie, F. K. Soong, and H. Li, "A KL divergence and DNN-based approach to voice conversion without parallel training sentences," in *INTERSPEECH*, 2016.

[6] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *ICASSP*, 2010.

[7] T. Kinnunen, L. Juvela, P. Alku, and J. Yamagishi, "Non-parallel voice conversion using i-vector PLDA: towards unifying speaker verification and transformation," in *ICASSP*, 2017.

[8] Y. Jia *et al.*, "Transfer learning from speaker verification to multi-speaker text-to-speech synthesis," in *NIPS*, 2018.

[9] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *ICLR*, 2014.

[10] C. Hsu *et al.*, "Voice Conversion from Non-parallel Corpora Using Variational Auto-Encoder," in *APSIPA*, 2016.

[11] C. Hsu *et al.*, "Voice Conversion from Unaligned Corpora Using Variational Autoencoding Wasserstein Generative Adversarial Networks," in *INTERSPEECH*, 2017.

[12] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," in *ICML*, 2017.

[13] L. Sun *et al.*, "Phonetic posteriorgrams for many-to-one voice conversion without parallel data training," in *ICME*, 2016.

[14] Y. Saito *et al.*, "Non-parallel voice conversion using variational autoencoders conditioned by phonetic posteriorgrams and d-vectors," in *ICASSP*, 2018.

[15] G. Zhao *et al.*, "Accent conversion using phonetic posteriorgrams," in *ICASSP*, 2018.

[16] G. Zhao *et al.*, "Foreign accent conversion by synthesizing speech from phonetic posteriorgrams," in *INTERSPEECH*, 2019.

[17] T. J. Hazen, W. Shen, and C. White, "Query-by-example spoken term detection using phonetic posteriorgram templates," in *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, 2009.

[18] X. Tian, E. S. Chng, and H. Li, "A Vocoder-free WaveNet Voice Conversion with Non-Parallel Data," *arXiv preprint arXiv:1902.03705*, 2019.

[19] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A Universal Music Translation Network," in *ICLR*, 2019.

[20] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural Discrete Representation Learning," in *NIPS*, 2017.

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

[22] R. Collobert *et al.*, "Wav2Letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[23] Y. Adi, N. Zeghidour, R. Collobert, N. Usunier, V. Liptchinsky, and G. Synnaeve, "To reverse the gradient or not: An empirical comparison of adversarial and multi-task learning in speech recognition," in *ICASSP*, 2019.

[24] 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*, 2016.

[25] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR corpus based on public domain audio books," in *ICASSP*, 2015.

[26] 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.

[27] H. Zen *et al.*, "LibriTTS: A corpus derived from LibriSpeech for Text-to-Speech," *arXiv preprint arXiv:1904.02882*, 2019.

[28] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders," in *ICML*, 2017.

[29] Y.-C. Wu *et al.*, "The NU non-parallel voice conversion system for the voice conversion challenge 2018," in *Odyssey*, 2018.

[30] C. Veaux, *et al.*, "CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit," *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, 2017.

[31] Google Cloud TTS robot, "US-WaveNet-E," https://cloud.google.com/text-to-speech/, 2018.

[32] J. Lorenzo-Trueba *et al.*, "The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods," *arXiv preprint arXiv:1804.04262*, 2018.

[33] F. P. Ribeiro, D. Florencio, C. Zhang, and M. Seltzer, "CROWDMOS: an approach for crowdsourcing mean opinion score studies," in *ICASSP*, 2011.

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

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

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

[37] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *ICASSP*, 2014.

[38] M. Ravanelli, T. Parcollet, and Y. Bengio, "The pytorch-kaldi speech recognition toolkit," in *ICASSP*, 2019.

[39] J. S. Garofolo *et al.*, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.