Conditional End-to-End Audio Transforms

Albert Haque[1*], *Michelle Guo*[1*], *Prateek Verma*[2]

[1]Department of Computer Science [2]Center for Computer Research in Music and Acoustics
Stanford University, USA

Abstract

We present an end-to-end method for transforming audio from one style to another. For the case of speech, by conditioning on speaker identities, we can train a single model to transform words spoken by multiple people into multiple target voices. For the case of music, we can specify musical instruments and achieve the same result. Architecturally, our method is a fully-differentiable sequence-to-sequence model based on convolutional and hierarchical recurrent neural networks. It is designed to capture long-term acoustic dependencies, requires minimal post-processing, and produces realistic audio transforms. Ablation studies confirm that our model can separate speaker and instrument properties from acoustic content at different context sizes. Empirically, our method achieves competitive performance on community-standard datasets.

1. Introduction

Humans are able to seamlessly process different audio representations despite syntactic, acoustic, and semantic variations. Inspired by humans, modern machine translation systems often use a word-level model to aid in the translation process [1, 2]. Many of these models are being used to model dependencies in music and speech for applications such as learning a latent space for representing speech, for text to speech and speech to text systems [3, 4]. In the case of text-based translation, learned word vectors or one-hot embeddings are the primary means of representing natural language [5, 6]. For speech and acoustic inputs however, word or phone embeddings are often used as a training convenience to provide multiple sources of information gradient flow to the model [7, 8]. Spectrograms remain the dominant acoustic representation for both phoneme and word-level tasks since the high sampling rate and dimensionality of waveforms is difficult to model [9].

In this paper, we address the task of end-to-end audio transformations, distinct from speech recognition (speech-to-text) and speech synthesis (text-to-speech). Specifically, we propose a fully-differentiable audio transformation model. Given an audio input (e.g. spoken word, single music note) conditioned on the speaker or instrument, the model learns to predict a output spectrogram for any arbitrary target speaker or instrument. Our framework is generic and can be applied across multiple applications including timbre transfer, accent transfer, speaker morphing, audio effects, and emotion transformation.

Current models often require complex pipelines consisting of domain-specific or fine-tuned features [10]. By contrast, our model does not require hand-crafted pipelines and only requires *conditioning* on input-output types. We evaluate our model on two tasks: (i) transforming words spoken by a human into multiple target voices, and (ii) playing a note on an instrument and transforming the note into another instrument's while retaining the pitch. We would like to make a key point: We do not supply the pitch variation, spectral or energy envelop to our model. Similarly for the output audio, parameters for synthesis are not provided but are instead learned from the desired mapping.

Our method operates on spectrograms and takes inspiration from Listen-Attend-Spell [11] and TacoTron [4, 12], and is combined with ideas from natural language processing [13]. Our contributions are two fold:

1. We present a fully differentiable end to end pipeline to perform audio transformations by conditioning on specific inputs and outputs for applications in speech and music.
2. We demonstrate that our model's learned embeddings produce a meaningful feature representation directly from speech. Our model transforms a spectrogram into a target spectrogram, with the only supervision being the input speaker identity and the requested target speaker. Our method has the flexibility to capture long term dependencies present in audio.

1.1. Related Work

Voice Conversion. Our work is related to the problem of voice conversion (VC) [14, 15]. Several works have approached VC using statistical methods based on Gaussian mixture models, which typically involves using parallel data [16]. Other works have also used neural network-based frameworks using restricted Boltzmann machines or feed-forward neural networks [17]. While most VC approaches require parallel source and target speech data, collecting parallel data can be expensive. Thus, few works have proposed parallel-data-free frameworks [18]. In our proposed method, time-aligned source and target speech data is not a prerequisite.

Recently, generative adversarial networks (GANs) have shown promise in image generatation and more recently in speech processing [19]. We refer the reader to the voice conversion challenge [20] for a more complete survey of VC methods.

Style Transfer. Our task of transforming audio from one style to another is closely related to the task of style transfer [21]. In visual style transfer [21, 22], the computed loss is typically a linear combination of the style and content loss, ensuring that the output is semantically similar to the input, despite variations in color and texture. By contrast, our work directly computes the loss on the output and ground truth spectrograms.

There have been works on similar lines for audio domain. In [23], Verma et al. provided an additional loss term, specific for audio, to transform musical instruments. Recent work on speech texture generation [24] shows promise of similar ideas and techniques for learning an end to end completely differentible pipeline. In [25], the authors introduced a version of vector quantization using variational autoencoders, which learned a code for a particular speech utterance and were able to achieve voice conversion by passing on additional speaker cues. Devel-

[*] Equal contribution.

oped concurrently with our work, style tokens [26] and musical translation [27] show the capabilities of unsupervised learning in the audio domain.

Text-to-Speech. Also known as speech synthesis, text-to-speech (TTS) systems have just recently started to show promising results. It has been shown that a pre-trained HMM combined with a sequence-to-sequence model can learn appropriate alignments [28]. Unfortunately this is not end-to-end as it predicts vocoder parameters and it is unclear how much performance is gained from the HMM aligner. Char2wav [29] is another method, trained end-to-end on character inputs to produce audio but also predicts vocoder parameters. These models show promise of capturing semantic, speaker and word level information in an small latent space which can be used for conditioning the text in a sequence to sequence decoder. DeepVoice [30] improves on this by replacing nearly all components in a standard TTS pipeline with neural networks. While this is closer towards a fully differentiable solution, each component is trained in isolation with different optimization objectives. [24] showed a fully differentiable end to end speech modification pipeline but the results were not convincing in terms of audio quality and was more to show a proof of concept.

WaveNet [9] is a powerful generative model of audio and generates realistic output speech. However it is slow due to audio sample-level autoregressive nature and requires domain-specific linguistic feature engineering. Our work take cues from the work done for personalization of chatbot response which condition the output of sequence to sequence models [13] to achieve consistent responses. We deploy a similar strategy by conditioning on the type of audio transformation we need for the input audio. Also similar is Tacotron line of work [4, 12]. In Tacotron [4], the authors move even closer to a fully differentiable system. The input to Tacotron [4] is a sequence of character embeddings and the output is a linear-scale spectrogram. After applying Griffin-Lim phase reconstruction [31], the waveform is generated.

2. Method

Our method is a sequence-to-sequence model [32] with attention. The encoder consists of a convolutional and pyramidal recurrent network [33]. The decoder is a recurrent network.

2.1. Encoder

Convolutional Network. Modeling the full spectrogram would require unrolling of the encoder RNN for an infeasibily large number of timesteps [34]. Even with truncated backpropagation through time [35], this would be a challenging task on large datasets. Inspired by the Convolutional, Long Short-Term Memory Deep Neural Network (CLDNN) [34] approach, we use a convolutional network to (i) reduce the temporal length of the input by using a learned convolutional filter bank. The stride, or hop size, controls the degree of length reduction. (ii) CNNs are good feature extractor that help the temporal unit better in modelling the longer dynamical features.

Pyramidal Recurrent Network. Inspired by the Clockwork RNN [33], we use a pyramidal RNN to address the issue of learning from a large number of timesteps [11]. A pyramidal RNN is the same as a standard multi-layer RNN but instead of each layer simply accepting the input from the previous layer, successively higher layers in the network only compute, or "tick," during particular timesteps. This allows different layers of the RNN to operate at different temporal scales. WaveNet [9]

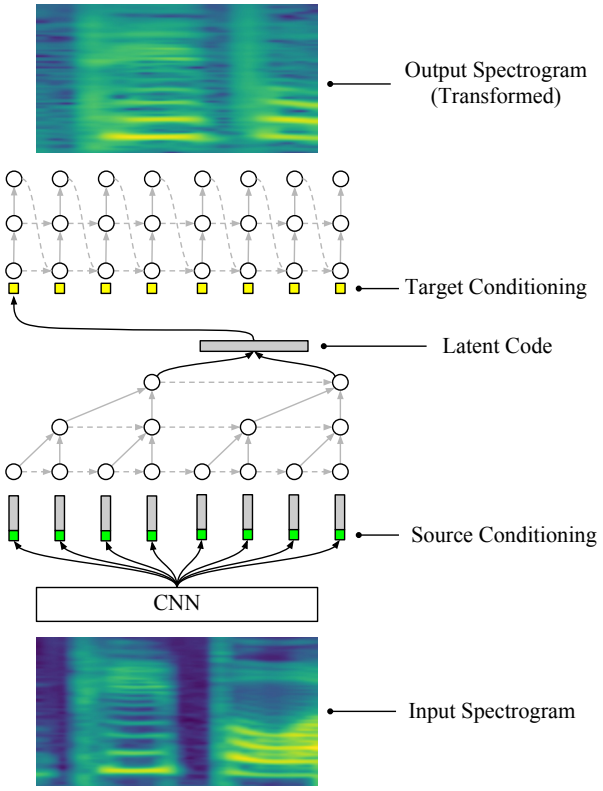

Figure 1: ***Overview of our model.*** *Yellow and green boxes denote one-hot vectors used to condition the input to each RNN. Solid lines indicate data flow. Dashed lines indicate temporal state sharing. Gray rectangles denote learned representations.*

also controls the temporal receptive field at each layer of their network with dilated convolutions [36, 37]. Formally, let h_i^j denote the hidden state of a long short-term memory (LSTM) cell at the i-th timestep of the j-th layer: $h_i^j = \text{LSTM}(h_{i-1}^j, h_i^{j-1})$ For a pyramidal LSTM (pLSTM), the outputs from the immediately preceding layer, which contains high-resolution temporal information, are concatenated:

$$h_i^j = \text{pLSTM}\left(h_{i-1}^j, \left[h_{2i}^{j-1}, h_{2i+1}^{j-1}\right]\right). \quad (1)$$

In (1), the output of a pLSTM unit is now a function of not only its previous hidden state, but also the outputs from two timesteps from the layer below. Not only does the pyramidal RNN provide higher-level temporal features, but it also reduces the inference complexity. Only the first layer processes each input timestep as opposed to all layers. The input time slice into the encoder is conditioned on the speaker ID. This is done by concatenating a one-hot speaker encoding with the CNN output at each time step, before being fed into the recurrent network.

2.2. Decoder

Attention. Learning long-range temporal dependencies can be challenging [38]. To aid this process, we use an attention-based LSTM transducer [39, 40]. At each timestep, the transducer produces a probability distribution over the next spectrogram time-slice conditioned on all previously seen inputs. The distribution for y_i is a function of decoder state s_i and context c_i.

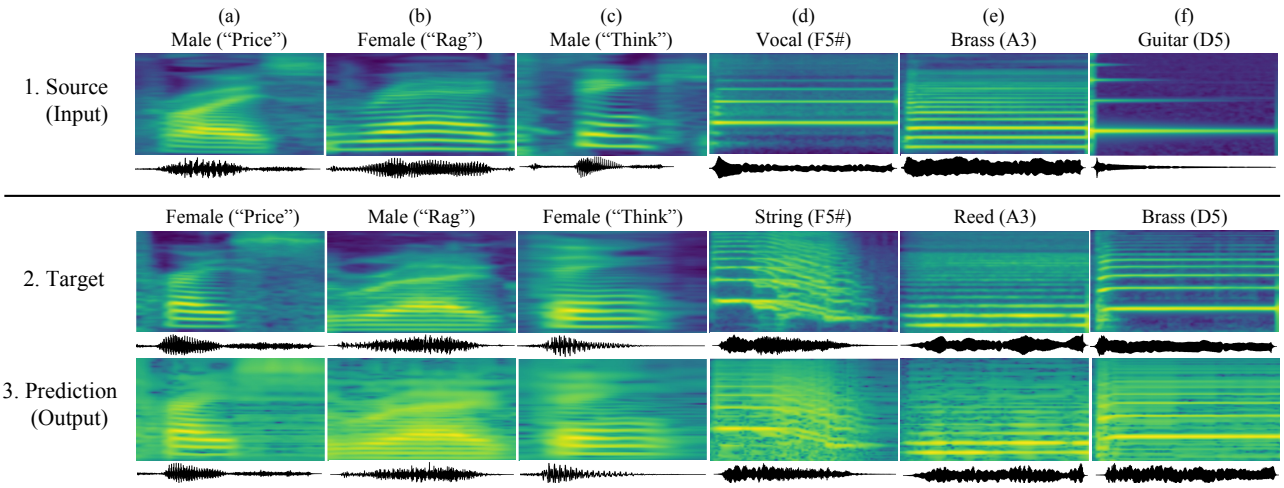

Figure 2: ***Spectrogram results.*** *Examples on TIMIT (2a-c) and NSynth (2d-f). The speakers and instruments were present in the training set but the test set contained unseen vocabulary words or notes. The x axis denotes time and the y axis denotes frequency. The input, ground truth target, and our model's prediction are shown. The corresponding waveform is shown beneath each spectrogram.*

The decoder state s_i is a function of previous state s_{i1}, previously emitted time-slice y_{i1}, context c_{i1} and one hot encoding of the desired output transformation similar to [13].

The context vector c_i is produced by an attention mechanism [11]. Specifically, we define:

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{j=1}^{L} \exp(e_{i,j})} \quad \text{and} \quad c_i = \sum_u \alpha_{i,u} h_u \qquad (2)$$

where attention $\alpha_{i,j}$ is defined as the alignment between the current decoder time-slice i and a time-slice j from the encoder input. The score between the output of the encoder (i.e., hidden states), h_j, and the previous state of the decoder cell, s_{i-1} is computed with: $e_{i,u} = \langle \phi(s_i), \varphi(h_u) \rangle$ where ϕ and φ are multi-layer perceptrons: $e_{i,j} = w^T \tanh(W s_{i-1} + V h_j + b)$ with learnable parameters w, W and V.

3. Experiments

Datasets. We evaluate our model on two audio transformation tasks: (i) voice conversion and (ii) musical style transfer. The TIMIT [41] and NSynth [42] datasets were used (Table 1). Speech examples from AudioSet [43] were used to pre-train the model as an autoencoder. We condition our model on the source and target style. For the case of speech, style refers to the speaker. For NSynth, it refers to the instrument type.

For all experiments, we focus on transformations on a word- or pitch-level. This was primarily for demonstration – adding sentence level transformations would have limited the number of training examples. However, we note that sequence-to-sequence models are capable of encoding sentence level information in a small latent encoding vector [11]. Similarly, the decoder can model full sentences, as shown in Tacotron [4].

Audio Format. All experiments used a sampling rate of 16 kHz with pre-emphasis of 0.97. Audio spectrograms were computed with Hann windowing, 50 ms window size, 12.5 ms hop size, and a 2048-point Fourier transform. Mel-spectrograms were visualized using an 80 channel mel-filterbank.

Optimization. The model was optimized with Adam [44] with $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The mean squared error was used as the loss objective. A learning rate of 10^{-3} was used with an exponential decay factor of 0.99 after every epoch. The batch size for all datasets was 128. Models were trained for 20 epochs on NSynth and 50 epochs on TIMIT. To train the decoder, we apply the standard maximum-likelihood training procedure, also known as teacher-forcing [45], which has been shown to improve convergence. The model was implemented and trained with Tensorflow on a single Nvidia V100 GPU.

Table 1: ***Overview of datasets.*** *We evaluate our method on voice conversion and musical style transfer.*

	TIMIT [41]	NSynth [42]	AudioSet [43]
Styles	630 speakers	1,006 types	7 classes
Content	6,102 words	88 pitches	—
Train	39,834	289,205	1,010,480
Test	14,553	4,096	—

Baselines. We evaluate three methods for audio transformations. First, we evaluate the classical sequence-to-sequence (Seq2Seq) model [32] which consists of a vanilla RNN as the encoder and a different RNN as the decoder. Second, we evaluate Listen, Atten, and Spell (LAS) [11]. This is the same as the seq2seq model but the decoder is equipped with an attention mechanism that allows it to “peek” at the encoder outputs. Additionally they use a pyramidal RNN. Third, we evaluate a conditional sequence-to-sequence model (C-Seq2Seq). This is the same as Seq2Seq but with our conditioning mechanism.

Evaluation Metrics. We use subjective and objective metrics for both voice conversion and musical style transfer:

- Mean opinion score (MOS), higher is better.
- Mel-cepstral distortion (MCD), lower is better.
- Side-by-side rating, higher is better.

Mel-Cepstral Distortion. Let y and $\hat{y}$ denote the ground truth and predicted mel-spectorgram, respectively. The MCD is:

$$\text{MCD}(y, \hat{y}) = \frac{10}{\ln(10)} \sqrt{2 \sum_{t=1}^{T} ||y_t - \hat{y}_t||} \qquad (3)$$

where T is the number of timesteps and t is the timestep slice.

Table 2: ***Comparison to existing methods.*** *We measure mean opinion score (MOS) and mel-cepstral distortion (MCD) on voice conversion and musical style transfer. Higher MOS is better. Lower MCD is better. The 95% confidence interval for TIMIT MOS and MCD values are* ± 0.024 *and* ± 0.017*, respectively, and for NSynth,* ± 0.016 *and* ± 0.053*. C-Seq2Seq is a vanilla conditioned seq-to sequence model.*

Method	MOS		MCD	
	TIMIT	NSynth	TIMIT	NSynth
Ground Truth	4.65	4.16	—	—
Seq2Seq [32]	3.37	3.13	7.31	11.18
LAS [11]	3.52	3.23	7.40	11.24
C-Seq2Seq	3.50	3.36	7.26	10.81
Our Method	**3.88**	**3.43**	**6.49**	**10.35**

3.1. Results

We present results on novel vocabulary words and pitches. Figure 2 shows results on the test set, which contain words and pitches not present in the training set. Our model is able to capture fundamental phonetic properties of each speaker or instrument and apply these properties to novel words and pitches.

3.1.1. Mean Opinion Score

To evaluate generative models, subjective scores [12] or perceptual metrics [22] are often used. We follow this same procedure for audio generated by our model by randomly selecting a fixed set of 50 test set examples. Audio generated from all baseline models on this set were rated by at least three normal-hearing human raters. A total of 10 raters participated in the study and listened to the generated audio with the same over-ear headphones. The rating scale is a 5-point numeric scale: 1. bad, 2. poor, 3. fair, 4. good, and 5. excellent. Higher values are better. The results of this study are shown in Table 2. Also shown in Table 2 are the mel-cepstral distances.

3.1.2. Side-by-Side Evaluation

We also conducted a side-by-side evaluation between audio generated by our system and the ground truth. For each prediction-ground truth audio pair, we asked raters to give a score ranging from -1 (generated audio is worse than ground truth) to +1 (generated audio is better than ground truth). The mean score was -0.74 ± 0.22, where 0.22 denotes the 95% confidence interval. This indicates that raters have a preference towards the ground truth.

3.1.3. Learned Style Representations

Figure 3 shows the learned representations for the musical transformation task. Our model can successfully cluster sounds belonging to pitch classes, rather than individual frequencies. More interestingly, from an acoustic signals perspective, a musical octave is denoted by plus/minus 12, where 12 is the standard MIDI pitch number. MIDI pitch 74 is one octave above MIDI 62. The resulting pitch 74 is double the frequency of pitch 62. However, pitch 74 is closer to 62 than 67, despite 67 being closer to 64 in absolute pitch. This is because 62 and 74 have high amounts of harmonic overlap. For audio transforms, this confirms our model can learn acoustic attributes without explicit supervision.

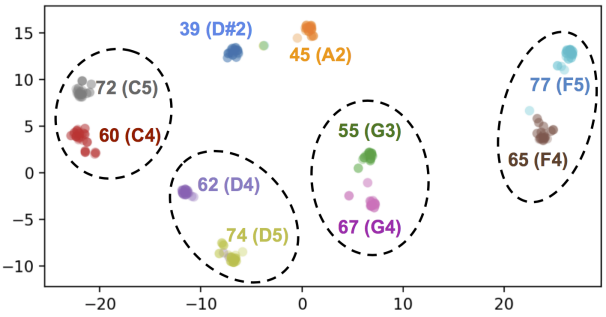

Figure 3: ***Learned representations.*** *Each dot denotes an audio clip. Colors denote pitches. The* x *and* y *axes are T-SNE [46] projections. Colored text denotes the musical pitch (e.g., D#2 refers to the note D#, 2nd octave). Dashed circles indicate pitch classes. Despite not given any explicit pitch labels, the model was able to cluster similar musical notes.*

3.1.4. Acoustic Context Size

Music demonstrates different acoustic properties compared to speech [47]. For example, speech often contains more variances in pitch whereas musical notes are held constant on the order of tens to hundreds of milliseconds. The acoustic context size denotes the contextual window modeled by each hidden state of our network. Larger contexts capture more acoustic content. As shown in Table 3, the context size does not significantly alter the mel-cepstral distance. All MCD values are between 10.3 and 10.7. Compare this to speech on TIMIT: a context size of 50 ms captures enough temporal context without overwhelming (i.e., smoothing) the hidden state. Larger fields of size 100-200 ms produce lower quality transforms.

Table 3: ***Acoustic context size.*** *We varied the per timestep context size and measured MCD. Lower values are better. Plus/minus values denote the 95% confidence interval.*

Context Size	TIMIT	NSynth
12.5 ms	7.28 ± 0.02	$\mathbf{10.35 \pm 0.05}$
25 ms	7.03 ± 0.02	10.47 ± 0.04
50 ms	$\mathbf{6.49 \pm 0.01}$	10.48 ± 0.05
100 ms	7.11 ± 0.02	10.69 ± 0.04
200 ms	7.32 ± 0.02	10.67 ± 0.05

4. Conclusion

In this work, we presented an end-to-end method for transforming audio from one style to another. Our method is a sequence-to-sequence model consisting of convolutional and recurrent neural networks. By conditioning on the speaker or instrument, our method is able to transform audio for unseen words and musical notes. Subjective tests confirm the quality of our model's generated audio. Overall, this work alleviates the need for complex audio processing pipelines and sheds new insights on the capabilities of end-to-end audio transformations. We hope others will build on this work and extend the capabilities of end-to-end audio transforms.

Acknowledgements. We would like to thank Malcolm Slaney and Dan Jurafsky for helpful feedback. Additionally, we thank members of the Stanford AI Lab for participating in subjective experiments.

5. References

[1] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv*, 2015.

[2] W. Y. Zou, R. Socher, D. M. Cer, and C. D. Manning, "Bilingual word embeddings for phrase-based machine translation." in *EMNLP*, 2013.

[3] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," *arXiv*, 2017.

[4] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv*, 2017.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, 2014.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv*, 2013.

[7] H. Bourlard and N. Morgan, "A continuous speech recognition system embedding mlp into hmm," in *NIPS*, 1990.

[8] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden markov models," *ICASSP*, 1990.

[9] 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*, 2016.

[10] A. Kain and M. W. Macon, "Spectral voice conversion for text-to-speech synthesis," in *ICASSP*. IEEE, 1998.

[11] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016.

[12] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *Interspeech*, 2018.

[13] J. Li, M. Galley, C. Brockett, G. P. Spithourakis, J. Gao, and B. Dolan, "A persona-based neural conversation model," *arXiv*, 2016.

[14] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara, "Voice conversion through vector quantization," *Journal of the Acoustical Society of Japan*, 1990.

[15] Y. Stylianou, O. Cappé, and E. Moulines, "Continuous probabilistic transform for voice conversion," *Trans. on speech and audio processing*, 1998.

[16] T. Toda, A. W. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory," *TASLP*, 2007.

[17] S. H. Mohammadi and A. Kain, "Voice conversion using deep neural networks with speaker-independent pre-training," in *Spoken Language Technology Workshop*, 2014.

[18] 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.

[19] T. Kaneko, H. Kameoka, K. Hiramatsu, and K. Kashino, "Sequence-to-sequence voice conversion with similarity metric learned using generative adversarial networks," in *Interspeech*, 2017.

[20] T. Toda, L.-H. Chen, D. Saito, F. Villavicencio, M. Wester, Z. Wu, and J. Yamagishi, "The voice conversion challenge 2016." in *Interspeech*, 2016.

[21] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *NIPS*, 2015.

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

[23] P. Verma and J. O. Smith, "Neural style transfer for audio spectograms," *arXiv*, 2018.

[24] J. Chorowski, R. J. Weiss, R. A. Saurous, and S. Bengio, "On using backpropagation for speech texture generation and voice conversion," *arXiv*, 2017.

[25] A. van den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," in *NIPS*, 2017.

[26] Y. Wang, D. Stanton, Y. Zhang, R. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, F. Ren, Y. Jia, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," *arXiv*, 2018.

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

[28] W. Wang, S. Xu, and B. Xu, "First step towards end-to-end parametric tts synthesis: Generating spectral parameters with neural attention," in *Interspeech*, 2016.

[29] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," *arXiv*, 2017.

[30] S. O. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, J. Raiman, S. Sengupta *et al.*, "Deep voice: Real-time neural text-to-speech," *arXiv*, 2017.

[31] S. Nawab, T. Quatieri, and J. Lim, "Signal reconstruction from short-time fourier transform magnitude," *TASLP*, 1983.

[32] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.

[33] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork rnn," *arXiv*, 2014.

[34] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Interspeech*, 2015.

[35] S. S. Haykin *et al.*, *Kalman Filtering and Neural Networks*. Wiley Online Library, 2001.

[36] P. Dutilleux, "An implementation of the algorithme à trous to compute the wavelet transform," in *Wavelets*. Springer, 1990.

[37] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv*, 2015.

[38] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Trans. on Neural Networks*, 1994.

[39] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015.

[40] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv*, 2014.

[41] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI*, 1993.

[42] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," 2017.

[43] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *ICASSP*, 2017.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[45] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, 1989.

[46] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, 2008.

[47] B. Gold, N. Morgan, and D. Ellis, *Speech and audio signal processing: processing and perception of speech and music*. John Wiley & Sons, 2011.