# FFTNET: A REAL-TIME SPEAKER-DEPENDENT NEURAL VOCODER

*Zeyu Jin*[1,2]*, Adam Finkelstein,*[1]

[1]Princeton University
Princeton, NJ 08540, USA

*Gautham J. Mysore*[2]*, Jingwan Lu,*[2]

[2]Adobe Research
San Francisco, CA 94103, USA

## ABSTRACT

We introduce FFTNet, a deep learning approach synthesizing audio waveforms. Our approach builds on the recent WaveNet project, which showed that it was possible to synthesize a natural sounding audio waveform directly from a deep convolutional neural network. FFTNet offers two improvements over WaveNet. First it is substantially faster, allowing for real-time synthesis of audio waveforms. Second, when used as a vocoder, the resulting speech sounds more natural, as measured via a "mean opinion score" test.

***Index Terms***— FFTNet, WaveNet, neural networks, vocoder

## 1. INTRODUCTION

The WaveNet project introduced a deep learning architecture capable of synthesizing realistic sounding human speech based on linguistic features and F0 pitch [1]. Though it was surprising to many researchers that a plausible waveform could be synthesized directly as the output of a convolutional neural network, it has subsequently been confirmed by many follow-on projects. The approach has many applications, including the classical text-to-speech (TTS) problem [2]. While WaveNet and others initially addressed TTS starting from *linguistic features,* ensuing work showed that speech could be synthesized directly from input *text* [3, 4]. The approach has also been adapted to other problems, including voice conversion [5, 6], speech enhancement [7], and musical instrument synthesis [1, 8].

Despite the impressive quality of the synthesized waveform, the WaveNet approach still suffers from several drawbacks: it requires substantial training corpus (roughly 30 hours), the synthesis process is slow (40 minutes to produce a second of audio), and the result contains audible noise. Recent work by Tamamori et al. [9] showed that WaveNet could also be used as a vocoder [10], which generates a waveform from *acoustic features*. Working from acoustic features, the training process is effective with a substantially smaller corpus (roughly one hour) [11] while still producing higher quality speech than baseline vocoders like MLSA [12]. Several research efforts have addressed the problem of computational cost. Paine et al. [13] introduce an algorithmic improvement for the same architecture called Fast WaveNet, which can synthesize a second of audio in a roughly a minute. The Deep Voice approach of Arik et al. [14] is able to achieve real-time synthesis by reducing the WaveNet model size significantly, but at the expense of noticeably reduced voice quality. Concurrent with our work, the WaveNet team also achieved huge performance gains by introducing a new deep learning network architecture that leverages parallelism on a GPU cluster [15].

This paper introduces FFTNet, an alternative deep learning architecture, coupled with several improved techniques for training and synthesis. WaveNet downsamples audio via dilated convolution in a process that resembles wavelet analysis. In contrast the FFTNet architecture resembles the classical Fast Fourier Transform (FFT) [16], and uses substantially fewer parameters than the analogous WaveNet model. FFTNet models produce audio more quickly ($> 70\times$ faster) than the Fast WaveNet formulation [13], thereby enabling real-time synthesis on a single CPU. Moreover, we show that when used as a vocoder, FFTNet produces higher quality synthetic voices, as measured by a "mean opinion score" test [17]. We also show that the FFTNet training and synthesis techniques can improve the original WaveNet approach such that the quality of the synthesized voice is on par with that of the FFTNet architecture (albeit much slower to synthesize). Finally, we note that the FFTNet architecture could also be leveraged in a variety of other deep learning problems such as classification tasks and autoencoders.

## 2. METHOD

Similar to WaveNet, our method generates waveforms one sample at a time based on previously generated samples and an auxiliary condition, but has a simpler architecture. In this section, we first briefly introduce WaveNet, then the proposed architecture called FFTNet, and finally a set of training and synthesis techniques essential in building a high quality FFTNet vocoder.

### 2.1. WaveNet vocoder

WaveNet [1] is a neural network architecture that has been used in audio synthesis to predict one audio sample at a time based on previously generated samples and auxiliary conditions, such as a sequence of phonemes and fundamental frequencies ($F0$). The prediction is based on the posterior distribution of sample values quantized using $\mu$-law. When the auxiliary conditions are acoustic features, such as Mel Cepstral Coefficients (MCC) and pitch, WaveNet can be used as a Vocoder, which generates a waveform from these features. It makes no assumption on how speech is generated and has been shown to generate significantly more natural and clear speech than conventional vocoders.

The basic building block of WaveNet is dilated causal convolution [18]: given signal $f$, dilation $d$ and kernel $k$, dilated convolution at time step $t$ is defined as $(k *_d f)_t = \sum_i k_i f_{t-di}$. If we replace the multiplication between $k_i$ and $f_{t-di}$ with $1 \times 1$ convolution (`conv1x1`), we get a layer of 1-D dilated convolutional neural network, or DCNN. WaveNet is constructed by stacking a series of DCNN layers with exponentially increasing dilation factors for each subsequent layer: $2^0, 2^1, 2^2, ...2^n$. At each layer, a gated activation structure is used:

$$z = \tanh(W_f *_d x + V_f *_d h) \odot \sigma(W_g *_d x + V_g *_d h)$$

where $x$ is the output from the previous layer and serves as the input for the current layer; $W_f$ and $W_g$ are convolution kernels for the

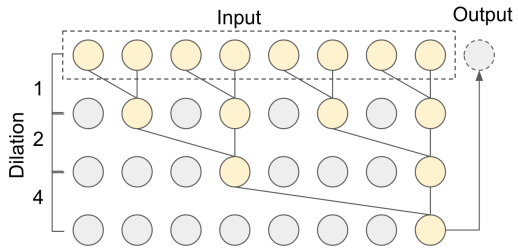



**Fig. 1**. Dilated convolution in WaveNet

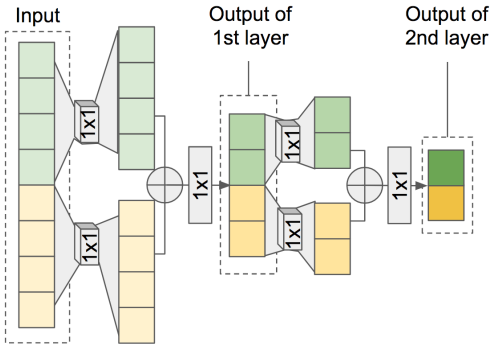



**Fig. 2**. FFTNet architecture: given size-8 inputs, they are first divided into two halves; each passed through a different $1\times1$ convolution layer and then summed together. The summed size-4 output will pass through ReLU and then another $1\times1$ convolution and ReLU before repeating the the same operation.

filters and the gates; $V_f$ and $V_g$ represent `conv1x1` on the auxiliary conditions $h$ (F0 and MCC); and $\odot$ is element-wise dot product. The final output of a layer is obtained by another `conv1x1` on $z$. The original WaveNet proposed to use skip connections – an additional `conv1x1` is applied to $z$ to obtain a new output $s$; then $s$ of all layers are summed together to become a skip output. To further increase nonlinearity, more `conv1x1` and ReLU layers are applied on top of the skip output and finally a softmax layer is used to produce the posterior distribution of quantized sample values.

With dilated convolution, a n-layer network has a receptive field of $2^n$ meaning as many as $2^n$ previous samples can influence the synthesis of the current sample, which leads to superior synthesis quality. However, since WaveNet synthesizes one sample at a time, to generate one second of audio sampled at 16kHz, the causal dilated network needs to be applied 16,000 times. Faster methods have been proposed [13], which can produce 200 samples per second, but the performance is still far from real-time on personal computers. We aim to design a vocoder with a simpler (thus faster) yet equally powerful architecture.

### 2.2. FFTNet architecture

Highlighting the nodes that influence the prediction of the new sample, we see a reversed binary tree structure as shown in Figure 1. This dilated convolution structure resembles wavelet analysis - in each step filtering is followed by down-sampling. This observation inspired us to explore an alternative structure to wavelet based on the Cooley-Tukey Fast Fourier Transform (FFT) [16]. Given an input sequence $x_1, x_2, ..., x_n$, FFT computes the $k$-th frequency component $f_k$ from time-domain series $x_0, ..., x_{N-1}$ by

$$f_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i n k/N} = \sum_{n=0}^{N/2-1} x_{2n} e^{-2\pi i (2n) k/N} + \sum_{n=0}^{N/2-1} x_{2n+1} e^{-2\pi i (2n+1) k/N}$$

Denote $\sum_{n=0}^{N-1} x_n e^{-2\pi i n k/N}$ as $f(n, N)$ and the above equation can be simplified as

$$\begin{aligned} f(n, N) &= f(2n, N/2) + f(2n+1, N/2) \\ &= f(4n, N/4) + f(4n+1, N/4) \\ &+ f(4n+2, N/4) + f(4n+4, N/4) = ... \end{aligned}$$

One can think of $x_n$ as a node with $K$ channels (e.g. 256 quantization channels) and the term $e^{-2\pi i (2n) k/N}$ as a transformation function, then each term $f(n, N) = f(2n, N/2) + f(2n+1, N/2)$ is analogous to applying a transformation to previous nodes $x_{2n}$ and $x_{2n+1}$ and summing up their results. If we use `conv1x1` as the transformation, we can create a network that resembles the FFT as shown in Figure 2. In other words, given input $x_{0:N}$ defined as a 1D series $(x_0, x_1, ..., x_{N-1})$, each FFTNet layer clips the input into two halves ($x_L = x_{0:N/2}$ and $x_R = x_{N/2:N}$), performs separate `conv1x1` transformation to each half and then sums up the results:

$$z = W_L * x_L + W_R * x_R \quad (1)$$

where $W_L$ and $W_R$ are `conv1x1` weights for $x_L$ and $x_R$. Instead of using a gated activation structure, FFTNet uses a simple ReLU followed by a `conv1x1` to produce inputs for the next layer, namely $x = \texttt{ReLU}(\texttt{conv1x1}(\texttt{ReLU}(z)))$, which reduces computation cost. Stacking $n$ layers will give an input size of $2^n$. The auxiliary conditions are transformed by `conv1x1` and then added to $z$. i.e.,

$$z = (W_L * x_L + W_R * x_R) + (V_L * h_L + V_R * h_R) \quad (2)$$

where $h_L$ and $h_R$ are the two halves of the condition vector $h$ and $V_L$ and $V_R$ are `conv1x1` weights. Note that if the condition information is stationary along the time axis, one may use $V * h_N$ instead of $(V_L * h_L + V_R * h_R)$.

To use FFTNet as a vocoder, we define $h_t$ as F0 and MCC features at time $t$. To generate the current sample $x_t$, we use the previously generated samples $x_{t-N:t}$ and auxiliary condition $h_{t-N+1:t+1}$ (shifted forward by 1) as the network input. In our experiments, the auxiliary condition is obtained as follows: first we take an analysis window of size 400 every 160 samples. Then we extract the MCC and F0 features for each overlapping window. For the $h_t$ corresponding to the window centers, we assign the computed MCC and F0 values (26 dimensions in total). For the $h_t$ that are not located at the window centers, we linearly interpolate their values based on the assigned $h_t$ in the last step.

Finally, FFTNet uses a fully connected layer followed by a softmax layer (size 1 with $K = 256$ channels) as the last two layers to produce the posterior distribution of the new sample's quantized values. To determine the final value of the current sample, one can use $argmax$ or perform random sampling from the posterior distribution, but we propose an alternative strategy in Section 2.3.2.

### 2.3. Training Techniques

In this section, we introduce a set of techniques that are essential for training an FFTNet to produce results of similar quality to WaveNet. Our experiments indicate that these techniques can also improve WaveNet synthesis quality.

#### 2.3.1. Zero padding

WaveNet uses zero-padding during dilated convolution. The same idea can be applied to FFTNet. Given a sequence of length $M$, the input $x_{1:M}$ ($M > N$) is shifted to the right by $N$ samples with zero padding. The $N$ padded zeros are denoted as $x_{-N:0}$ where $\forall j < 0, x_j = \mathbf{0}$. Equation 1 becomes

$$z_{0:M} = W_L * x_{-N:M-N} + W_R * x_{0:M}$$

Our experiments show that without zero padding, the network tends to produces noise or gets stuck (outputting zeros) when the inputs are nearly silent. Zero-padding during training allows the network to generalize to partial input. We recommend using training sequences of length between $2N$ and $3N$ so that a significant number (33% - 50%) of training samples are partial sequences. In this work, we apply zero-padding to both WaveNet and FFTNet in our experiment.

#### 2.3.2. Conditional sampling

Both WaveNet and FFTNet are classification networks - the last softmax layer produces a posterior distribution over 256 categories. Like every classifier, the prediction error comes from two sources: training error and true error. True error mainly corresponds to noise, and resides in the unvoiced parts of the signal. To synthesize noise, we rely on the network to learn the noise's distribution by the output posterior distribution on which we use random sampling to obtain the sample's value. Training error comes from the model itself; and the prediction strategy that gives the minimal training error is $argmax$. However, $argmax$ is not suitable for simulating signals that contain true noise, since it always chooses the center of a noise distribution leading to zero noise in the synthesis output. Instead of using $argmax$ universally, we propose to use different prediction strategies for unvoiced and voiced sounds (Figure 3). For unvoiced sounds, we randomly sample from the posterior distribution; and for voiced sounds, we take the normalized logits (the input values before softmax), multiply it by a constant $c > 1$ and pass it through the softmax layer to obtain a posterior distribution where random sampling is performed. In this way, the posterior distribution will look steeper while the original noise distribution is preserved. In this work, we use $c = 2$.

#### 2.3.3. Injected noise

Due to the training error, the synthesized samples always contain some amount of noise; during synthesis, the network will generate samples that get noisier over time. They serve as network input to generate the next sample, adding more and more randomness to the network. When the noise builds up, the output sample might drift leading to clicking artifacts. To avoid such drift, the network needs to be robust to noisy input samples. We achieve this by injecting random noise to the input $x_{0:M}$ during training. We determine the amount of noise to inject into the input based on the amount of noise the network is likely to produce. In this work, we observe that the prediction is often one category (out of 256) higher or lower than the ground-truth category and thus, we inject Gaussian noise centered at 0 with a standard deviation of 1/256 (based on 8 bit quantization).

#### 2.3.4. Post-synthesis denoising

Our experiments show that the injected noise eliminates the clicking artifact almost perfectly for FFTNet but introduces a small amount of random noise to voiced samples. Therefore, we apply a spectral subtraction noise reduction [19] to reduce the injected noise for the voiced samples. The reduction is proportional to the amount of noise injected during training. It is possible to apply noise reduction to the unvoiced samples as well, but it may result in artifacts. Therefore, we reduce the amount of noise reduction applied to the unvoiced samples by half.

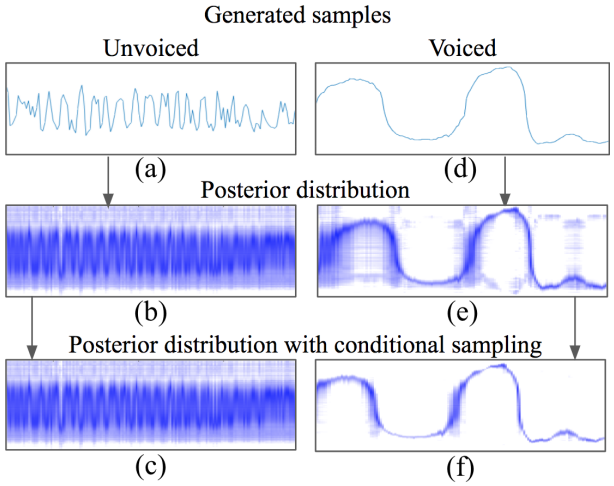


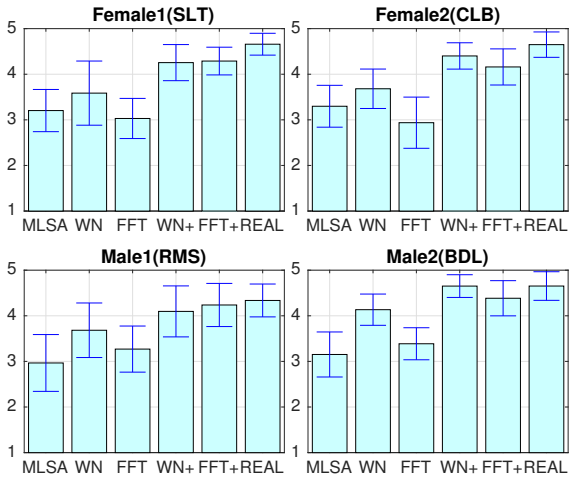

**Fig. 3**. Conditional sampling: WaveNet and FFTNet model noise by matching the posterior distribution to the noise's distribution (b). Therefore to generate noise (a), we randomly sample from the posterior distribution (b). For periodic signals (d), we double the log of the posterior distribution (e), to obtain a cleaner distribution (f); for aperiodic signals, the posterior distribution remains the same (c).

## 3. EVALUATION

To evaluate our algorithm, we conduct both a subjective listening test using mean-opinion score (MOS) to measure synthesis quality, and an objective test to measure spectral and cepstral distortion.

### 3.1. Experimental setup

Four voices, two male (BDL,RMS) and two female (SLT,CLB), from the CMU Arctic dataset [20] are used in our experiments. The first 1032 utterances (out of 1132) are used for training and the remaining are used for evaluation. The waveforms are quantized to 256 categorical values based on $\mu$-law. 25-coefficient Mel Cepstral Coefficients (with energy) and F0 are extracted from the original samples. We build 4 networks for each voice, 2 WaveNets and 2 FFTNets. For each type of network, we used two training strategies: Strategy one is with zero padding but without using the other techniques in Section 2.3; Strategy two applies all techniques in Section 2.3. The WaveNet we implemented contains two stacks of 10-layer dilated convolution ($d = 2^0, 2^1, ..., 2^9$) with 256 dilation and 128 skip channels. The total receptive field is 2048 samples. We experimented with different numbers of channels and found the current configuration the best for the vocoder. The FFTNet implementation contains 11 FFT-layers with 256 channels, which also has a receptive field of 2048. Note that this FFTNet has less than 1M parameters and with proper caching [13], the computation cost for generating one second of audio (16kHz) is only around 16GFLOPs; it means a modern CPU can generate audio samples in real-time. In each training step, a minibatch of $5\times$ 5000-sample sequences are fed to the network, optimized by the Adam algorithm [21] with a training rate $0.001$. We set the variance of injected noise to be $1/256$. In each minibatch, all sequences come




| | MLSA | WN | FFT | WN+ | FFT+ | Real |
|---|---|---|---|---|---|---|
| MOS | 3.16 | 3.77 | 3.16 | 4.35 | 4.27 | 4.57 |

**Fig. 4**. Mean opinion score (MOS) test results show that the proposed method FFT+ improves synthesis quality over the original WaveNet WN, and that our training/synthesis techniques (+) improve both original WaveNet and naive FFTNet (FFT). The bottom table shows the average MOS across four voices.

from different utterances; we trained WaveNet by 200,000 steps and FFTNet by 100,000 steps to ensure convergence.

In our implementations, synthesis using FFTNet is more than 70 times faster than Fast WaveNet [13], requiring only 0.81 second to generate one second of audio on a laptop CPU (2.5 GHz Intel Core i7). The audio clips and results of the experiments described in this section may be found at our project web page.[1]

### 3.2. Subjective evaluation

We conducted a Mean Opinion Score (MOS) test [17] that asks subjects to rate the quality of the synthetic utterances. Our subjects were recruited via Amazon Mechanical Turk (AMT), a micro-task platform popular for crowdsourcing experiments [22]. We recruited participants from the United States who have an approval rate over 90% to ensure the reliability of the study results. We also designed a validation test to ensure that subjects are paying attention and rejected those who fail on those tests. AMT has been used for conducting MOS tests in various domains [23, 24, 25]. It has been shown to be effective for various kinds of listening tests [26, 27] and can provide a large number of diverse participants in a short amount of time. We evaluated six conditions for each utterance:

| | |
|---|---|
| MLSA | MLSA filter |
| WN | WaveNet with only zero-padding |
| FFTN | FFTNet with only zero-padding |
| WN+ | WaveNet with techniques in Section 2.3 |
| FFTN+ | FFTNet with techniques in Section 2.3 |
| Real | the actual recording |

In each task (called a *HIT*), a subject is presented with 32 different sentences in which 24 of them are made of 4 instances from each of the above 6 conditions. From a held-out set of sentences, we add 4 more instances of the "Real" condition and 4 more cases of badly edited "Fake" (3 bit A-law encoded) condition to validate that the subject is paying attention and not guessing randomly. For the data to be retained, the subject may make at most one mistake on these validation tests, by either rating $< 3$ on "Real" examples or $> 3$ on "Fake" examples. We launched 480 HITs (120 per voice) and retained 446 after validation.

Figure 4 shows a bar chart for the MOS test with the error bars indicating standard deviation across utterances. The proposed training technique improves both WaveNet and FFTNet significantly with an ANOVA test $p$-value less than $10^{-9}$ for both networks. The proposed network FFT+ also improves on WN with a $p$-value of $< 10^{-20}$. Both WN+ and FFT+ have significant overlap with the real examples in MOS scores. The proposed method FFT+ has a slightly lower MOS than WaveNet WN+ (with an insignificant $p$-value); but it is significantly faster, as noted above. It is also worth noting that FFT has similar quality to the baseline method MLSA (insignificant $p$-value) due to noisy artifacts; this implies the training and synthesis techniques are essential to make FFTNet work well.

### 3.3. Objective evaluation

We evaluated the distortion between the original and the synthesized speech using RMSE and MCD [9]. RMSE measures frequency domain difference between two signals; and MCD measures the difference in the cepstral domain, which reflects whether the synthesized speech can capture the characteristics of the original speech. Both measurements are in dB. The result is shown in Table 1.

| | MCD (dB) | | | | RMSE (dB) | | | |
|---|---|---|---|---|---|---|---|---|
| voice | slt | clb | rms | bdl | slt | clb | rms | bdl |
| mlsa | 2.76 | 3.03 | 3.62 | 3.28 | 8.05 | 9.14 | 8.80 | 8.25 |
| WN | 4.47 | 4.04 | 4.60 | 3.05 | 9.71 | 9.65 | 9.38 | 8.29 |
| WN+ | 4.57 | 4.13 | 4.41 | 3.28 | 9.80 | 8.95 | 9.74 | 8.67 |
| FFT | 5.24 | 5.07 | 4.82 | 4.23 | 10.39 | 9.77 | 10.33 | 10.13 |
| FFT+ | 4.73 | 4.69 | 4.41 | 3.82 | 9.88 | 9.58 | 9.89 | 9.64 |

**Table 1**. Comparison of distortion between natural speech and synthetic speech based on Average MCD and RMS

The result shows that MLSA tends to preserve most of the cepstral and spectral structure while the MOS test puts it in a significantly lower tier as it generates audible over-smoothing artifacts. The techniques introduced in Section 2.3 do not reduce distortion in WaveNet but they significantly improve FFTNet in both metrics. Also, note that the WaveNet with the proposed techniques performs significantly better in subjective evaluation than the one without. This result also contradicts MCD and RMSE.

## 4. CONCLUSION

We introduce FFTNet, a neural network that generates waveforms one sample at a time based on previously generated samples and axillary conditions. Unlike WaveNet, FFTNet uses a simple architecture mimicking the Fast Fourier Transform (FFT) that makes it possible to generate audio samples in real-time. We also propose a set of training and synthesis techniques that improve the synthesis quality of both FFTNet and WaveNet. Experiments demonstrate that when used as a vocoder, FFTNet generates higher-quality speech than the original WaveNet, and on par with WaveNet as improved by our training and synthesis techniques. Due to its relatively small number of parameters, FFTNet is not currently suited for speaker-independent vocoding. Future work will consider application of this architecture to other problems such as phoneme classification and acoustic modeling.

[1] http://gfx.cs.princeton.edu/pubs/Jin_2018_FAR

## 5. REFERENCES


[1] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[2] Thierry Dutoit, *An introduction to text-to-speech synthesis*, vol. 3, Springer Science & Business Media, 1997.

[3] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous, "Tacotron: A fully end-to-end text-to-speech synthesis model," *CoRR*, vol. abs/1703.10135, 2017.

[4] Jose Sotelo, Soroush Mehri, Kundan Kumar, Joao Felipe Santos, Kyle Kastner, Aaron Courville, and Yoshua Bengio, "Char2wav: End-to-end speech synthesis," in *ICLR 2017 workshop*, 2017.

[5] Miguel Varela Ramos, "Voice conversion with deep learning," *Tecnico Lisboa Masters thesis*, 2016.

[6] Kazuhiro Kobayashi, Tomoki Hayashi, Akira Tamamori, and Tomoki Toda, "Statistical voice conversion with wavenet-based waveform generation," *Proc. Interspeech 2017*, pp. 1138–1142, 2017.

[7] Dario Rethage, Jordi Pons, and Xavier Serra, "A wavenet for speech denoising," *CoRR*, vol. abs/1706.07162, 2017.

[8] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," *arXiv preprint arXiv:1704.01279*, 2017.

[9] Akira Tamamori, Tomoki Hayashi, Kazuhiro Kobayashi, Kazuya Takeda, and Tomoki Toda, "Speaker-dependent wavenet vocoder," in *Proceedings of Interspeech*, 2017, pp. 1118–1122.

[10] Homer Dudley, "The vocoder – electrical re-creation of speech," *Journal of the Society of Motion Picture Engineers*, vol. 34, no. 3, pp. 272–278, 1940.

[11] Alan W Black, Heiga Zen, and Keiichi Tokuda, "Statistical parametric speech synthesis," in *Proc. ICASSP*, 2007, vol. 4, pp. IV–1229.

[12] Satoshi Imai, Kazuo Sumita, and Chieko Furuichi, "Mel log spectrum approximation (mlsa) filter for speech synthesis," *Electronics and Communications in Japan (Part I: Communications)*, vol. 66, no. 2, pp. 10–18, 1983.

[13] Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A Hasegawa-Johnson, and Thomas S Huang, "Fast wavenet generation algorithm," *arXiv preprint arXiv:1611.09482*, 2016.

[14] Sercan O Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Jonathan Raiman, Shubho Sengupta, et al., "Deep voice: Real-time neural text-to-speech," *arXiv preprint arXiv:1702.07825*, 2017.

[15] Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis, "Parallel wavenet: Fast high-fidelity speech synthesis," *CoRR*, vol. abs/1711.10433, 2017.

[16] James W Cooley and John W Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[17] Anderson F Machado and Marcelo Queiroz, "Voice conversion: A critical survey," *Proc. Sound and Music Computing (SMC)*, pp. 1–8, 2010.

[18] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves, "Conditional image generation with pixelcnn decoders," in *NIPS*, 2016, pp. 4790–4798.

[19] Philipos C Loizou, *Speech enhancement: theory and practice*, CRC press, 2013.

[20] John Kominek and Alan W Black, "The cmu arctic speech databases," in *Fifth ISCA Workshop on Speech Synthesis*, 2004.

[21] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling, "Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data?," *Perspectives on psychological science*, vol. 6, no. 1, pp. 3–5, 2011.

[23] Maxine Eskenazi, Gina-Anne Levow, Helen Meng, Gabriel Parent, and David Suendermann, *Crowdsourcing for speech processing: Applications to data collection, transcription and assessment*, John Wiley & Sons, 2013.

[24] Zeyu Jin, Adam Finkelstein, Stephen DiVerdi, Jingwan Lu, and Gautham J Mysore, "Cute: A concatenative method for voice conversion using exemplar-based unit selection," in *Proc. ICASSP*, 2016, pp. 5660–5664.

[25] Zeyu Jin, Gautham J. Mysore, Stephen Diverdi, Jingwan Lu, and Adam Finkelstein, "Voco: Text-based insertion and replacement in audio narration," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 96:1–96:13, July 2017.

[26] Mark Cartwright, Bryan Pardo, Gautham J Mysore, and Matt Hoffman, "Fast and easy crowdsourced perceptual audio evaluation," in *Proc. ICASSP*, 2016, pp. 619–623.

[27] Jeanne Parson, Daniela Braga, Michael Tjalve, and Jieun Oh, "Evaluating voice quality and speech synthesis using crowdsourcing," in *International Conference on Text, Speech and Dialogue*. Springer, 2013, pp. 233–240.