Vol.2019-SLP-126 No.3
2019/2/27

Investigating neural source-filter waveform model for statistical parametric speech synthesis

Xin Wang[1,a)] Shinji Takaki[1,b)] Junichi Yamagishi[1,c)]

Abstract: Recently we proposed the neural source-filter model (NSF) that converts a sequence of acoustic features into a speech waveform. Similar to other recent neural waveform models, the NSF is a non-autogressive model powered by dilated CNN; however, the NSF uses the sine waveform instead of the random noise as the excitation. Furthermore, without using the normalizing flow, the NSF simply optimizes the network parameters by minimizing a spectral amplitude distance. In this work, we further investigated the three issues: whether the network structure can be further simplified; whether the NSF can be applied to multi-speaker speech synthesis; whether the NSF can be directly applied to convert the linguistic features into the speech waveforms. Our experiments showed positive results on all the three points. Particularly, we found that the WaveNet-style gated activation can be safely removed, and the NSF performs quite well as a pure dilated-CONV-based network.

Keywords: neural waveform model, text-to-speech

1. Introduction

Neural-network-based waveform models [1], [2], [3] are essential components for recent text-to-speech (TTS) systems. As Figure 1 shows, a neural waveform model can be used to convert the predicted acoustic features into the waveform or generate the waveform directly from the linguistic features.

One of the primary neural waveform model called the WaveNet [1] generates the waveform sampling point one by one. Unfortunately, the autoregressive (AR) waveform generation process of the WaveNet is annoyingly time-consuming. There have been two strategies for fast neural waveform generation. First, based on the AR WaveNet, the WaveRNN was proposed to accelerate the waveform generation speed using engineering hacks [4], e.g., simultaneous generation of multiple waveform sampling points. Another strategy for fast generation is to switch the AR model to the inverse autoregressive flow (IAF) [5], which allows all the waveform sampling points to generated at the same time [2], [3]. However, these IAF-based models must be trained given the 'knowledge' provided by a pre-trained WaveNet. A recent model called WaveGlow [6] further relieves the IAF-based models of the knowledge-distilling training approach.

Different from the above approaches, we have proposed a neural waveform model with neither the AR nor the IAF model assumption. This proposed model, which is referred to as a neural source filter waveform (NSF) model [7], uses a source module to generate sine-wave-based excitation signals with a specified fundamental frequency (F0). It then transforms the excitation signal into the speech waveform through multiple network blocks based on dilated convolution (CONV) layers. To ensure that such as a straightforward model can learn the complicated waveform distributions, we further proposed a short-time-Fourier-transform (STFT)-based training criterion. Different from the commonly-used cross-entropy or mean-square-error calculated over each waveform sampling point individually, the STFT-based training criterion evaluates the spectral amplitude distances between the generated and natural waveforms in short-time analysis windows. Our experiments have demonstrated that the NSF model was similar to the AR WaveNet in terms of the quality of synthetic speech. Meanwhile, the NSF was at least 100 times faster [7].

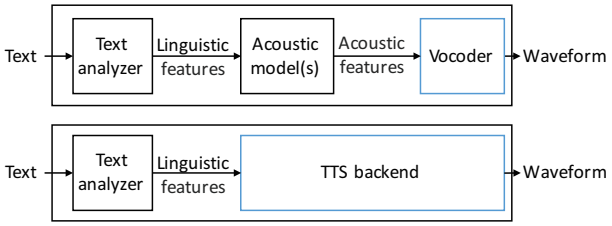

Fig. 1 Pipeline TTS architectures. Neural waveforms models can be used as a vocoder (figure above) or a unified TTS back-end (figure below).

In this technical report, we investigate the NSF model in three aspects: whether the NSF model can be further simplified; whether the NSF model supports multi-speaker waveform modeling; whether the NSF model can be applied to generated waveforms from linguistic features rather than acoustic features. Our experiments showed positive results. First, the NSF model can get rid of the WaveNet-style gated activation components. Based on this structure, we found that the NSF can be directly trained on a multi-speaker corpus without using speaker codes. Finally, it is also possible to train a NSF-based TTS backend to convert linguistic features into the waveform, even though the resulting waveforms lack spectral details in the high-frequency band.

[1] National Institute of Informatics, Tokyo, Japan
[a)] wangxin@nii.ac.jp
[b)] takaki@nii.ac.jp
[c)] jyamagish@nii.ac.jp

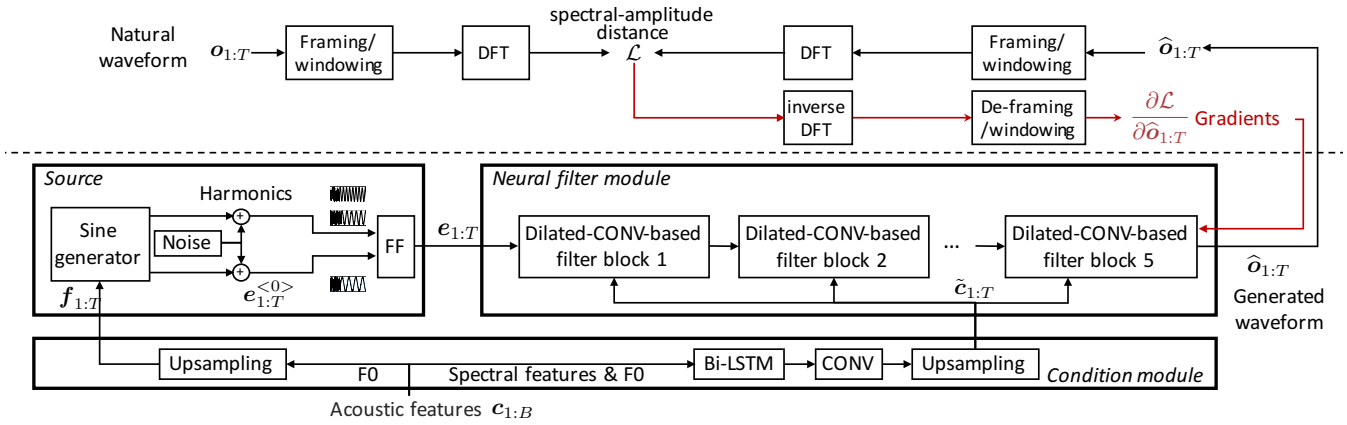

Fig. 2 General structure of NSF model, where B and T denote lengths of input feature sequence and output waveform, respectively. FF, CONV, and Bi-LSTM denote feedforward, convolution, and bi-directional recurrent layers, respectively. DFT denotes discrete Fourier transform. Dilated-CONV-based filter blocks have the same network structure and will be explained in Figure 3.

The original and the simplified NSF model will be explained in Section 2 and 3, respectively. After that, the application of the simplified NSF in TTS will be explained in Section 4.

2. Original NSF model

The original NSF model is plotted in Figure 2. It generally consists of three modules: a condition module that processes and upsamples the frame-level input features $\boldsymbol{c}_{1:B}$, a source module that generates a sine waveform with the specified fundamental frequency (F0) and the corresponding harmonics, a dilated-CONV-based filter module that converts the excitation $\boldsymbol{e}_{1:T}$ into a speech waveform $\widehat{\boldsymbol{o}}_{1:T}$. In addition to the three modules, the original NSF model is trained using the STFT-based training criterion [*1].

Except the filter module, other components of the original NSF model are straightforward to implement. The upsampling layer in the condition module directly duplicate the frame-level input features. For example, if the acoustic features are extracted with a frame shift of 5ms (i.e., 200Hz) while the waveform has a sampling rate of 16kHz, each input feature vector is repeated 80 times ($80 = 16000/200$) so that the upsampled feature sequence $\tilde{\boldsymbol{c}}_{1:T}$ has the same length as that of the waveform.

Given the upsampled F0 sequence $\boldsymbol{f}_{1:T}$, the source module generates a sine waveform that carries the F0 as

$$e_t^{<0>} = \begin{cases} \alpha \sin(\sum_{k=1}^{t} 2\pi \frac{f_k}{N_s} + \phi) + n_t, & \text{if } f_t > 0 \\ \frac{1}{3\sigma} n_t, & \text{if } f_t = 0 \end{cases}, \qquad (1)$$

where $n_t \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian noise, $\phi \in [-\pi, \pi]$ is a random initial phase, and N_s is equal to the waveform sampling rate. Note that $f_t > 0$ denotes being voiced while $f_t = 0$ denotes being unvoiced. Harmonics can be generated by simply multiply f_k with an integer. The $e_t^{<0>}$ and the harmonics are then merged through a feedforward layer into the one-dimensional excitation signal $e_{1:T}$. Note that the scaler α is used to adjust the signal amplitude so that it has roughly the same maximum amplitude in voiced and unvoiced segments. Here we use $\alpha = 0.1$ and $\sigma = 0.003$.

The dilated-CONV-based filter module in the original NSF model is very similar to that used in the original WaveNet [*2]. It includes multiple yet same structured dilated-CONV-based blocks, among which the structure of a single block is plotted at the top of Figure 3. In each block, the one-dimensional input signal $\boldsymbol{x}_{1:T}$ is first expanded in dimension by using a feedforward layer. After that, it is processed by a dilated CONV layer, merged with the upsampled conditional feature $\tilde{\boldsymbol{c}}_{1:T}$, and split into two parts before passing through the tanh and sigmoid activation functions. The output of the tanh and sigmoid functions are merged through element-wise product and transformed by feedorward layers. Note that all the hidden feature sequences have the same time length T but different dimensions.

After repeating this process, some of the hidden feature sequences can be propagated by the skip-channel (orange links) to the output side of the block and transformed into one-dimensional signals $\boldsymbol{a}_{1:T}$ and $\boldsymbol{b}_{1:T}$. Finally, the input signal $\boldsymbol{x}_{1:T}$ is scaled and shifted as the output signal $\boldsymbol{y}_{1:T} = \boldsymbol{x}_{1:T} \odot \boldsymbol{b}_{1:T} + \boldsymbol{a}_{1:T}$. Note that the dilation size of the k-th dilated conv layer in a single block is $2^{(k-1)}$. The dimensions of the skip-channel and residual channel are 128 and 64, respectively.

As Figure 2 illustrates, each dilated-CONV-based processing block generates two signals $\boldsymbol{b}_{1:T}$ and $\boldsymbol{a}_{1:T}$ given the input signal. It then transforms the input signal through a simple affine transformation, which is identical to that used in ClariNet. However, because the NSF doesn't use the IAF framework, at each time step t, b_t and a_t can be generated using the information from the input signal not only before but also after the time step t. In other words, there is no need to use causal dilated CONV layers to ensure the causal dependency between $\{a_t, b_t\}$ and x_t [*3].

3. Simplified NSF model

The network structure of the original NSF model was designed

[*1] Details of the STFT-based training criterion and gradients computation have been explained in the appendix of the arxiv paper https://arxiv.org/abs/1810.11946

[*2] Details of the WaveNet can be found in http://tonywangx.github.io/pdfs/wavenet.pdf

[*3] In the our previous work [7] and this work, however, we still use the causal dilated CONV layers so that we can make a fair comparison between the NSF and the WaveNet.

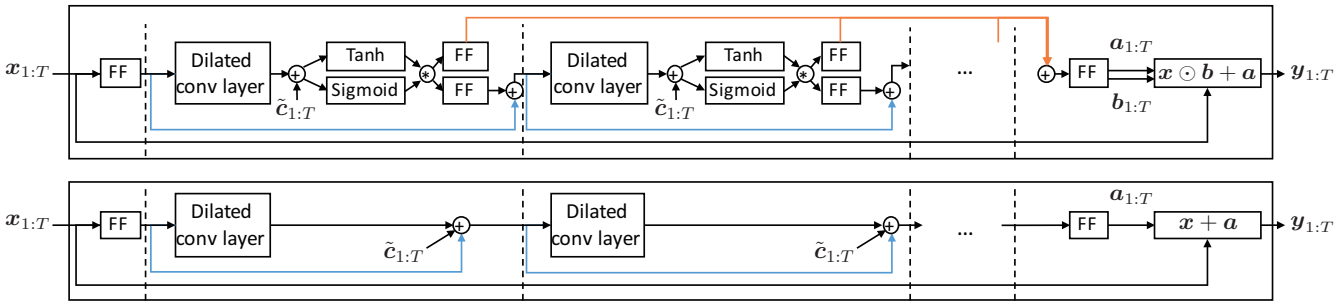

Fig. 3 Details of a single dilated-CONV-based processing block in the original NSF model (above) and the simplified NSF model (bottom). The orange links denote the skip-channels, and the blue links denote the residual channel. $\tilde{c}_{1:T}$ denote the feature sequence given by the conditional module.

so that we can compare the original NSF with the WaveNet fairly. However, each dilated-CONV-based filter block can be simplified. Through trials and errors, we found that the structure at the bottom of Figure 3 is sufficient.

First, the input one-dimensional signal $\boldsymbol{x}_{1:T}$ is expanded in dimension using a feedforward layer. After it is propagated through the dilated convolution layer, the dimension-expanded signal is summed with the output of the dilated CONV layer and the up-sampled features $\tilde{\boldsymbol{c}}_{1:T}$. Of course, this summation requires that $\tilde{\boldsymbol{c}}_{1:T}$ and the input and the output of the dilated CONV layer have the same dimension. For this purpose, we set the dimension of $\tilde{\boldsymbol{c}}_t$ and the size of the feedforward and the dilated CONV layer to be 64. The simple combination of dilated CONV and summation is repeated 10 times in a single processing block as Figure 3 sjpws. The output of the last summation operator is transformed into the one-dimensional signal $\boldsymbol{a}_{1:T}$ using a feedforward layer, and the $\boldsymbol{a}_{1:T}$ is added to the $\boldsymbol{x}_{1:T}$ as the output of the whole processing block $\boldsymbol{y}_{1:T}$.

Compared with the original NSF, the simplified version uses a simpler neural filter module. Because the filter module is the stem of the NSF model, a simpler filter module means that the time to generate the waveform and the size of the whole NSF model could be intensively reduced.

This is demonstrated by the generation speed and the network size listed in Table 1 and 2. Note that the implementation of `NSF` has a normal and a memory-save generation mode. In the normal mode, the implementation allocates all the required GPU memory to generate the waveforms. Obviously, the required memory would increase as the waveform becomes longer. To alleviate the memory requirement, the memory-save mode releases and allocates the memory layer by layer. However, the repeated memory operations cost additional processing time. For the test in Table 1, we generated waveforms around 5s in length so that the NSF models can work in a normal mode on our GPU card. Of course, the simplified NSF could generate longer waveforms in the normal mode since it requires less GPU memory.

On the quality of the generated waveforms, we perceived almost the same level of high quality when we compared the original and the simplified NSF models. Future work will conduct a formal listening test.

Table 1 Average number of waveform points generated in 1s, tested on a single Nvidia P100 GPU card.

`WaveNet`	`Original NSF`		`Simplified NSF`	
	memory-save	normal	memory-save	normal
0.19k	20k	227k	88k	327k

Table 2 Number of network weights

`WaveNet`	`Original NSF`	`Simplified NSF`
$2.9e+6$	$1.8e+6$	$1.1e+6$

4. Application of NSF model for TTS

In our previous work [7], we mainly use the NSF to replace the traditional vocoder in a speaker-dependent classical pipeline-based TTS system. In other words, the input of the NSF is assumed to be the acoustic features for a specific speaker, including the Mel-generalized-cepstral (MGC) coefficients and the F0. However, it is flexible to change the input features and apply the NSF model for other TTS applications.

4.1 Multi-speaker waveform modeling

One of the important TTS application is the multi-speaker TTS, i.e., generating natural speech waveforms that convey the speaker identity of a specified speaker. In the conventional pipeline TTS using a classical signal-processing-based vocoder, the speech waveform of a specific speaker can be well generated as long as the input acoustic features contain the speaker information. However, a neural waveform model must be test to show that it can learn from a multi-speaker corpus and preserve each speaker's identity in the generated speech waveforms.

4.1.1 System building

For the above purpose, we built a multi-speaker TTS system on the VCTK corpus [8], using the NSF model as the vocoder. The VCTK corpus contains utterances from around 108 non-professional English speakers. We randomly selected around 200 utterances from 68 speakers as the training set and built the TTS system following the common recipe: first, acoustic features including the MGC and F0 were extracted with a frame shift of 5ms and a frame length of 20 ms. Then, the linguistic features were derived using the front-end of Festival and aligned with the acoustic features using HTK toolkit [*4]. Given the aligned linguistic, acoustic features, and one-hot speaker codes, neural-network-

[*4] We used the script from the Merlin speech synthesis toolkit [9]

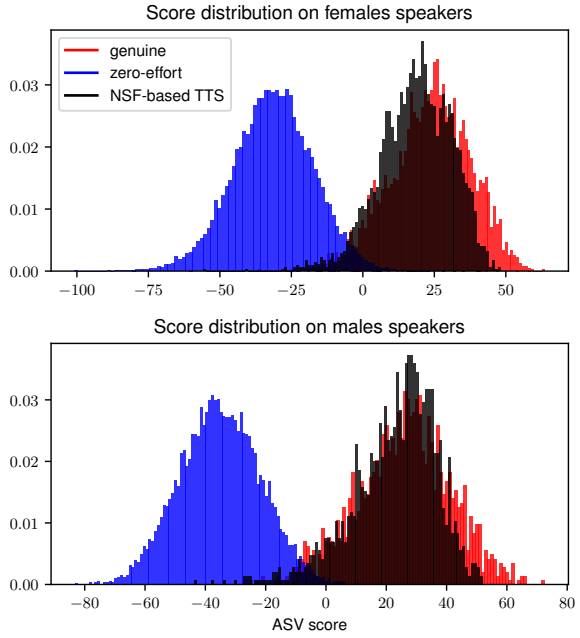

Fig. 4 Distributions of scores evaluated by the ASV system on the genuine, zero-effort imposter, and the NSF-based TTS imposter. Scores on the female (above) and male (below) speakers are plotted.

based duration model and acoustic models were trained. Both duration and acoustic models used simple neural networks with two feedforward and two Bi-LSTM layers. The NSF model were directly trained on the natural acoustic features and waveforms without speaker-codes.

4.1.2 Evaluation

The trained TTS system is expected to be able to generate the waveforms for each speaker. To test the degree of speaker identity preserved in the generated waveforms, we evaluated these generated waveforms with help of a GMM-UBM-based automatic-speaker-verification (ASV) system [10]. The goal of an ASV system is to decide whether an input speech waveform is uttered by a target speaker, and its decision is mainly determined by the speaker similarity between the input speech waveform and the waveform enrolled by the target speaker. We use this ASV system to evaluate whether the generated waveforms of our TTS system sound similar to the corresponding speakers [*5].

Among the 68 speakers, we randomly selected 48 speakers as the target speakers. First, 25 natural non-training utterances were selected for each of the target speaker as the enrolling utterance. After enrolling, another about 100 natural utterances from each target speaker were used as the genuine waveforms. For the rest of the 20 speakers, we selected their natural utterances as the zero-effort imposter waveforms. Meanwhile, we used the trained TTS system to generate the waveforms for the 48 target speakers and collected them as the TTS-based imposter waveforms.

By evaluating the three groups of waveforms using the ASV system, we gathered scores and plotted the score's distribution for each group. Note that the data of female and male speakers were separately evaluated. As Figure 4 plotted, the distribution of the zero-effort imposter waveforms is well separated from that of the genuine waveforms, which means that the ASV system well believes that most of zero-effort waveforms do not sound similar to any of the target speaker. However, the distribution of the TTS-based imposter waveforms is highly overlapped with that of the genuine speech, which indicates a high degree of speaker similarity between the TTS-generated speech and the genuine speech. Table 3 further plots the ASV equal error rate averaged over the target speakers. Note that a perfect imposter would result in an equal error rate of 50%.

[*5] Using the TTS output to attack ASV systems have risen intensive security concerns. To address this issue, researchers have been working on anti-spoofing systems that protect the ASV systems by detecting TTS generated waveforms from natural waveforms [11].

Table 3 Equal error rate

Zero-effort imposter		Simplified NSF	
Male	Female	Male	Female
1.85%	2.48%	46.36%	40.24%

The above results indicate that the TTS-based system can generate speech waveforms that sound close to the target speaker. Because the TTS-based system uses the NSF model to generate the waveform, we can infer that the NSF model indeed preserved the speaker identity even though it was trained by simply pooling the data from multiple speakers without using speaker codes.

4.1.3 Waveform generation for unknown speakers

Another multi-speaker TTS application is to generate the speech waveform for an unknown speaker. Being unknown means that the speaker is not included in the training data. A preliminary test showed that the NSF can directly generate good waveforms given the acoustic features from an unknown speaker. Even the NSF trained on the VCTK corpus can be used to generate the speech of an unknown speaker speaking in Japanese. We leave the examination on the universality of the NSF model to the future work.

4.2 From linguistic features to waveform

The last experiment on the NSF model is to covert the linguistic features into the speech waveforms. In this sense, the NSF model is a TTS-backend that combines the acoustic model and the neural waveform model. To better extract the hidden features from the input linguistic features, we used a conditional module that had two feedforward layers of size 512, two Bi-LSTM layers of size 256, and an output layer of size 64.

Figure 5 plots the spectrograms of the natural speech, generated speech from the NSF vocoder given natural/predicted acoustic feature, and the generated speech from the NSF TTS-backend. The natural F0 was used for all the NSF models. In perception, the NSF TTS-backend can generate understandable waveforms, but these waveforms sound muffled. One reason is the over-smoothed high-frequency band in the generated waveforms. This is expected because the linguistic features cannot encode the details of the waveforms. The ambiguity between the linguistic features and waveforms lead to the over-smoothing effect, especially on the high-frequency band.

5. Conclusion

The original NSF model is a promising model for waveform modeling, but it unnecessarily complicated in terms of model structure. This report explains how the NSF model can be sim-

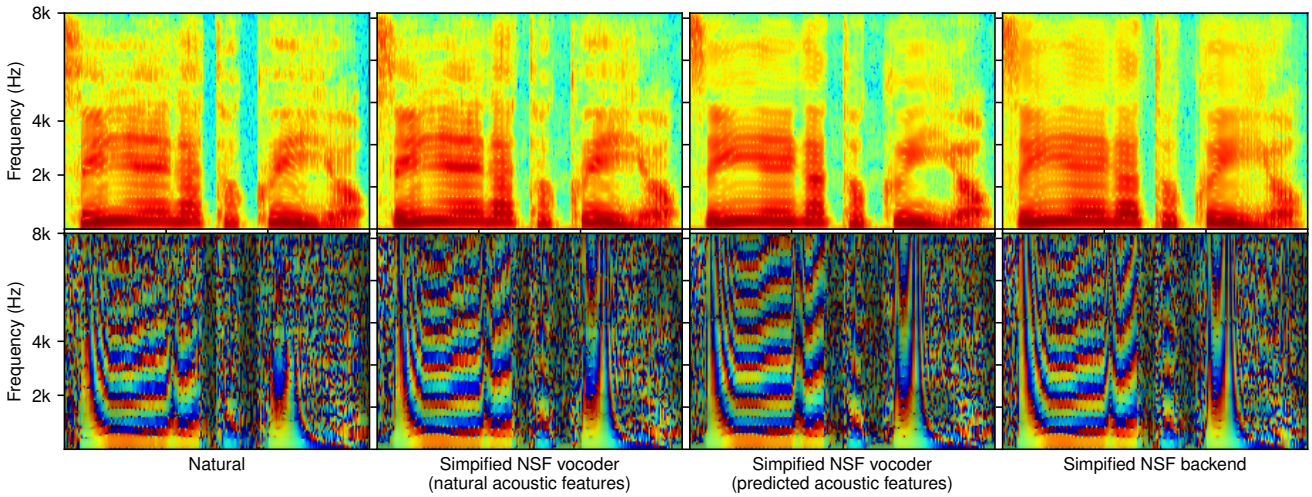

Fig. 5 Spectrogram (row above) and instantaneous frequency (row blow) of natural waveform, generated waveform from simplified NSF with natural acoustic feature as input and generated waveform from simplified NSF with linguistic features as input.

plified. By removing the gated activation function and other unnecessary hidden layers in the filter module, the simplified NSF is lighter, faster, and more debugging-friendly. Despite the simple model structure, the NSF can be directly trained on a multi-speaker corpus. It is also possible to use the NSF to directly convert linguistic features into the waveform.

References

[1] van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D. and Hassabis, D.: Parallel WaveNet: Fast High-Fidelity Speech Synthesis, *Proc. ICML*, pp. 3918–3926 (2018).

[2] Oord, A. v. d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G. v. d., Lockhart, E., Cobo, L. C., Stimberg, F. et al.: Parallel WaveNet: Fast high-fidelity speech synthesis, *arXiv preprint arXiv:1711.10433* (2017).

[3] Ping, W., Peng, K. and Chen, J.: ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech, *arXiv preprint arXiv:1807.07281* (2018).

[4] Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., van den Oord, A., Dieleman, S. and Kavukcuoglu, K.: Efficient Neural Audio Synthesis, *Proc. ICML* (Dy, J. and Krause, A., eds.), Proceedings of Machine Learning Research, Vol. 80, Stockholmsmässan, Stockholm Sweden, PMLR, pp. 2410–2419 (online), available from ⟨http://proceedings.mlr.press/v80/kalchbrenner18a.html⟩ (2018).

[5] Tomczak, J. M. and Welling, M.: Improving Variational Auto-Encoders using convex combination linear Inverse Autoregressive Flow, *Proc. Benelearn*, pp. 162–194 (2017).

[6] Prenger, R., Valle, R. and Catanzaro, B.: WaveGlow: A Flow-based Generative Network for Speech Synthesis, *Proc. ICASSP*, p. (to appear) (2019).

[7] Wang, X., Takaki, S. and Yamagishi, J.: Neural source-filter-based waveform model for statistical parametric speech synthesis, *Proc. ICASSP*, p. (to appear) (2019).

[8] Veaux, C., Yamagishi, J., MacDonald, K. et al.: CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (2017).

[9] Wu, Z., Watts, O. and King, S.: Merlin: An open source neural network speech synthesis system, *Proc. SSW, Sunnyvale, USA* (2016).

[10] Reynolds, D. A., Quatieri, T. F. and Dunn, R. B.: Speaker verification using adapted Gaussian mixture models, *Digital signal processing*, Vol. 10, No. 1-3, pp. 19–41 (2000).

[11] Kinnunen, T., Sahidullah, M., Delgado, H., Todisco, M., Evans, N., Yamagishi, J. and Lee, K. A.: The ASVSpoof 2017 challenge: Assessing the limits of replay spoofing attack detection (2017).