Edinburgh Research Explorer

Neural Source-Filter Waveform Models for Statistical Parametric Speech Synthesis

OPEN ACCESS

Neural source-filter waveform models for statistical parametric speech synthesis

Xin Wang, *Member, IEEE,* Shinji Takaki, *Member, IEEE,* Junichi Yamagishi, *Senior Member, IEEE*

Abstract—**Neural waveform models have demonstrated better performance than conventional vocoders for statistical parametric speech synthesis. One of the best models, called WaveNet, uses an autoregressive (AR) approach to model the distribution of waveform sampling points, but it has to generate a waveform in a time-consuming sequential manner. Some new models that use inverse-autoregressive flow (IAF) can generate a whole waveform in a one-shot manner but require either a larger amount of training time or a complicated model architecture plus a blend of training criteria.**

As an alternative to AR and IAF-based frameworks, we propose a neural source-filter (NSF) waveform modeling framework that is straightforward to train and fast to generate waveforms. This framework requires three components to generate waveforms: a source module that generates a sine-based signal as excitation, a non-AR dilated-convolution-based filter module that transforms the excitation into a waveform, and a conditional module that pre-processes the input acoustic features for the source and filter modules. This framework minimizes spectral-amplitude distances for model training, which can be efficiently implemented using short-time Fourier transform routines. As an initial NSF study, we designed three NSF models under the proposed framework and compared them with WaveNet using our deep learning toolkit. It was demonstrated that the NSF models generated waveforms at least 100 times faster than our WaveNet-vocoder, and the quality of the synthetic speech from the best NSF model was comparable to that from WaveNet on a single-speaker Japanese speech corpus.

Index Terms—**speech synthesis, neural network, waveform model, short-time Fourier transform**

I. INTRODUCTION

Text-to-speech synthesis (TTS), a technology that converts a text string into a speech waveform [1], has rapidly advanced with the help of deep learning. In TTS systems based on a statistical parametric speech synthesis (SPSS) framework [2], [3], deep neural networks have been used to derive linguistic features from text [4]. More breakthroughs have been reported for the acoustic modeling part, for which various types of neural networks have been proposed that convert linguistic features into acoustic features such as the fundamental fre-

Xin Wang is with National Institute of Informatics, Tokyo, 101-8340, Japan. e-mail:wangxin@nii.ac.jp

Shinji Takaki is with the Nagoya Institute of Technology, Japan. e-mail: takaki@sp.nitech.ac.jp

Junichi Yamagishi is with the National Institute of Informatics, Tokyo, 101-8340, Japan and also with Centre for Speech Technology Research, University of Edinburgh, UK. e-mail: jyamagis@nii.ac.jp.

quency (F0) [5], [6], cepstral or spectral features [7], [8], [9], and duration [10], [11].

Deep learning has recently been used to design a vocoder, a component that converts acoustic features into a speech waveform. The pioneering model called WaveNet [12] directly generates a waveform in a sample-by-sample manner given the input features. It has shown better performance than classical signal-processing-based vocoders for speaker-dependent SPSS systems [13]. Similar AR models such as SampleRNN [14] have also performed reasonably well for SPSS [15]. The key idea of these neural waveform models is to implement an autoregressive (AR) probabilistic model that describes the distribution of a current waveform sample conditioned on previous samples. Although AR models can generate high-quality waveforms, their generation speed is slow because they have to generate waveform samples one by one.

Inverse-AR is another approach to neural waveform modeling. For example, inverse-AR-flow (IAF) [16] can be used to transform a noise sequence into a waveform without the sequential generation process. However, each waveform must be sequentially transformed into a noise-like signal during model training [17], which significantly increases the amount of training time. Rather than the direct training method, Parallel WaveNet [18] and ClariNet [19] use a pre-trained AR model as a teacher to evaluate the waveforms generated by an IAF student model. Although this training method avoids sequential transformation, it requires a pre-trained WaveNet as the teacher model. Furthermore, additional training criteria have to be incorporated, without which the student model only produces mumbling sounds [18], [19]. The blend of disparate training criteria and the complicated knowledge-distilling approach make IAF-based frameworks even less accessible to the TTS community.

In this paper, we propose a neural source-filter (NSF) waveform modeling framework, which is straightforward to implement, fast in generation, and effective in producing high-quality speech waveforms for TTS. NSF models designed under this framework have three components: a source module that produces a sine-based excitation signal, a filter module that uses a dilated-convolution-based network to convert the excitation into a waveform, and a conditional module that processes the acoustic features for the source and filter modules. The NSF models do not rely on AR or IAF approaches and are directly trained by minimizing the spectral amplitude distances between the generated and natural waveforms.

We describe the three specific NSF models designed under our proposed framework: a baseline NSF (b-NSF) model that adopts a WaveNet-like filter module, simplified NSF (s-NSF)

model that simplifies the filter module of the baseline NSF model, and harmonic-plus-noise NSF (hn-NSF) model that uses separate source-filter pairs for the harmonic and noise components of waveforms. While we previously introduced the b-NSF model [20], the s-NSF and hn-NSF models are newly introduced in this paper. Among the three NSF models, the hn-NSF model outperformed the other two in a large-scale listening test. Compared with WaveNet, the hn-NSF model generated speech waveforms with comparably good quality but much faster.

We review recent neural waveform models in Section II and describe the proposed NSF framework and three NSF models in Section III. After explaining the experiments in Section IV, we draw a conclusion in Section V.

II. REVIEW OF RECENT NEURAL WAVEFORM MODELS

We consider neural waveform modeling as the task of mapping input acoustic features into an output waveform. Suppose one utterance has B frames and its acoustic features are encoded as a sequence $\boldsymbol{c}_{1:B} = \{\boldsymbol{c}_1, \cdots \boldsymbol{c}_B\}$, where $\boldsymbol{c}_b \in \mathbb{R}^D$ is the feature vector for the b-th frame. Without loss of generality, we define the target waveform as $\boldsymbol{o}_{1:T} = \{o_1, \cdots, o_T\}$, where $o_t \in \mathbb{R}$, and T is the waveform length[1]. We then use $\widehat{o}_{1:T}$ to denote the waveform generated by a neural waveform model.

A. Naive neural waveform model

Let us first consider a naive neural waveform model. After upsampling $\boldsymbol{c}_{1:B}$ to $\tilde{\boldsymbol{c}}_{1:T}$ by replicating each $\boldsymbol{c}_b$ multiple times, we may use a recurrent or convolution network to learn the mapping from $\tilde{\boldsymbol{c}}_{1:T}$ to $\boldsymbol{o}_{1:T}$. Such a model can be trained by minimizing the mean square error (MSE):

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \sum_{l=1}^{L} \sum_{t=1}^{T_l} (o_t^{(l)} - \widehat{o}_t^{(l)})^2, \qquad (1)$$

where $\widehat{o}_t^{(l)} = \mathcal{F}_{\boldsymbol{\Theta}}(\boldsymbol{c}_{1:T_l}^{(l)}, t)$ is the output of the neural network at the t-th time step for the l-th training utterance, and $\boldsymbol{\Theta}$ is the network's parameter set. During generation, the model can generate a waveform $\widehat{\boldsymbol{o}}_{1:T} = \{\widehat{o}_1, \cdots, \widehat{o}_t\}$ by setting $\widehat{o}_t = \mathcal{F}_{\boldsymbol{\Theta}^*}(\boldsymbol{c}_{1:T}, t), \forall t \in \{1, \cdots, T\}$ given the input $\boldsymbol{c}_{1:T}$.

However, such a naive model did not produce intelligible waveforms in our pilot test. We found that the generated waveform was over-smoothed and lacked the variations that evoke the perception of speech formant. This over-smoothing effect may be reasonable because the way we train the naive model is equivalent to maximizing the likelihood of a waveform $\boldsymbol{o}_{1:T}$ over the distribution

$$p(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:T}; \boldsymbol{\Theta})$$
$$= \prod_{t=1}^{T} p(o_t|\boldsymbol{c}_{1:T}; \boldsymbol{\Theta}) = \prod_{t=1}^{T} \mathcal{N}(o_t; \mathcal{F}_{\boldsymbol{\Theta}}(\boldsymbol{c}_{1:T}, t), \sigma^2), \qquad (2)$$

where $\mathcal{N}(\cdot; \cdot, \sigma^2)$ is a Gaussian distribution with an unknown standard deviation σ. It is assumed that the waveform values are independently distributed with the naive model, which may be incompatible with the strong temporal correlation of natural waveforms. This mismatch between model assumption and natural data distribution may cause the over-smoothing effect.

[1]Some waveform models require discretized waveform values, i.e., $o_t \in \mathbb{N}$.

B. Neural AR waveform models

AR neural waveform models have recently been proposed for better waveform modeling [12]. Contrary to the naive model assumption in Equation (2), it is assumed that

$$p(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:B}; \boldsymbol{\Theta}) = \prod_{t=1}^{T} p(o_t|\boldsymbol{o}_{<t}, \boldsymbol{c}_{1:B}; \boldsymbol{\Theta}) \qquad (3)$$

with an AR model, where $p(o_t|\boldsymbol{o}_{<t}, \boldsymbol{c}_{1:B}; \boldsymbol{\Theta})$ depends on the previous waveform values $\boldsymbol{o}_{<t}$. Such a model can potentially describe the causal temporal correlation among waveform samples.

An AR model can be implemented by feeding the waveform sample of the previous time step as the input to a convolution network (CNN) or recurrent network (RNN) at the current step, which is illustrated in Figure 1. The model is trained by maximizing the likelihood over natural data while using the data for feedback, i.e., teacher forcing [21]. In the generation stage, the model has to sequentially generate a waveform and use the previously generated sample as the feedback datum.

Pioneering neural AR waveform models include WaveNet [12] and SampleRNN [14]. There are also models that combine WaveNet with classical speech-modeling methods such as glottal waveform modeling [22] and linear-prediction coding [23], [24]. Although these models can generate high-quality waveforms, their sequential generation process is time-consuming. More specifically, these AR models' time complexity to generate a single waveform of length T is theoretically equal to $\mathcal{O}(T)$. Note that the time complexity only takes the waveform length into account, ignoring the number of hidden layers and implementation tricks such as subscale dependency [25] and squeezing [17]. Other models, such as WaveRNN [25], FFTNet [26], and subband WaveNet [27], use different network architectures to reduce or parallelize the computation load, but their generation time is still linearly proportional to the waveform length.

Note that in the teacher-forcing-based training stage, the CNN-based (e.g., WaveNet) and RNN-based AR models (e.g., WaveRNN) have time complexity of $\mathcal{O}(1)$ and $\mathcal{O}(T)$, respectively. The bottleneck of the RNN-based models is the computation in the recurrent layers. These theoretical interpretations are summarized in Table I.

C. IAF-flow-based models

Rather than the AR-generation process, an IAF-flow-based model such as WaveGlow [17] uses an invertible and non-AR function $H_{\boldsymbol{\Theta}}^{-1}(\cdot)$ to convert a random Gaussian noise signal $\widehat{\boldsymbol{z}}_{1:T}$ into a waveform $\widehat{\boldsymbol{o}}_{1:T} = H_{\boldsymbol{\Theta}}^{-1}(\widehat{\boldsymbol{z}}_{1:T}, \boldsymbol{c}_{1:B})$. In the training stage, the model likelihood has to be evaluated as

$$p_O(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:B}; \boldsymbol{\Theta})$$
$$= p_Z(\boldsymbol{z}_{1:T} = H_{\boldsymbol{\Theta}}(\boldsymbol{o}_{1:T}, \boldsymbol{c}_{1:B})) \left| \det \frac{\partial H_{\boldsymbol{\Theta}}(\boldsymbol{o}_{1:T}, \boldsymbol{c}_{1:B})}{\partial \boldsymbol{o}_{1:T}} \right| \qquad (4)$$

where $H_{\boldsymbol{\Theta}}(\boldsymbol{o}_{1:T}, \boldsymbol{c}_{1:B})$ sequentially inverts a training waveform $\boldsymbol{o}_{1:T}$ into a noise signal $\boldsymbol{z}_{1:T}$ for likelihood evaluation. When $H_{\boldsymbol{\Theta}}^{-1}(\cdot)$ is sufficiently complex, $p_O(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:B}; \boldsymbol{\Theta})$ can approximate the true distribution of $\boldsymbol{o}_{1:T}$ [28].
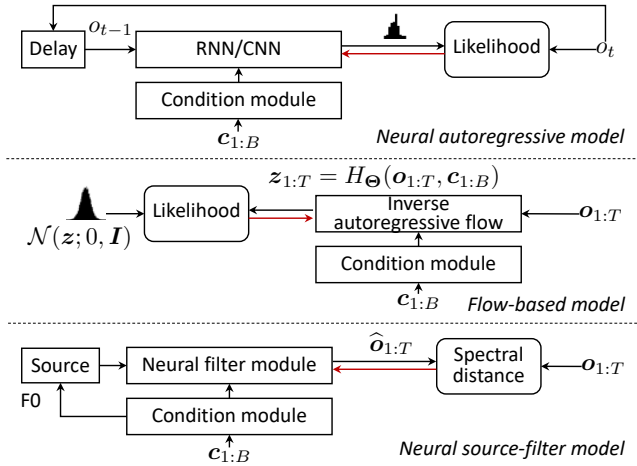
Fig. 1. Three types of neural waveform models in training stage. $\widehat{\boldsymbol{o}}_{1:T}$ and $\boldsymbol{o}_{1:T}$ denote generated and natural waveforms, respectively. $\boldsymbol{c}_{1:B}$ denotes input acoustic features. Red arrows denote gradients for back propagation.

TABLE I
THEORETICAL TIME COMPLEXITY TO PROCESS WAVEFORM OF LENGTH T

Model	Time complexity		Knowledge distilling
	train	generation	
AR (CNN-based, e.g., WaveNet)	$\mathcal{O}(1)$	$\mathcal{O}(T)$	×
AR (RNN-based, e.g., WaveRNN)	$\mathcal{O}(T)$	$\mathcal{O}(T)$	×
IAF flow (e.g., WaveGlow)	$\mathcal{O}(T)$	$\mathcal{O}(1)$	×
AR+IAF (e.g., Parallel WaveNet)	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✓
NSF	$\mathcal{O}(1)$	$\mathcal{O}(1)$	×

In terms of the theoretical time complexity w.r.t T, the flow-based models are dual to the CNN-based AR models, as Table I summaries. Although the time complexity of the flow-based models in waveform generation is irrelevant to T, their time complexity in model training is $\mathcal{O}(T)$. Some models such as WaveGlow reduce T by squeezing multiple waveform samples into one vector [17] but still require a huge amount of training time. For example, in one study using Japanese speech data [29], a WaveGlow model was trained on four V100 GPU cards for one month to produce high-quality waveforms[2].

D. AR plus IAF

Some models, such as Parallel WaveNet [18] and ClariNet [19], combine the advantages of AR and IAF-flow-based models under a distilling framework. In this framework, a flow-based student model generates waveform samples $\widehat{\boldsymbol{o}}_{1:T} = H^{-1}_{\boldsymbol{\Theta}, \boldsymbol{c}_{1:B}}(\widehat{\boldsymbol{z}}_{1:T})$ and queries an AR teacher model to evaluate $\widehat{\boldsymbol{o}}_{1:T}$. The student model learns by minimizing the distance between $p(\widehat{\boldsymbol{o}}_{1:T})$ and that given by the teacher model. Therefore, neither training nor generation requires sequential transformation.

However, it was reported that knowledge distilling is insufficient and additional spectral-domain criteria must be used [18], [19]. Among the blend of disparate loss functions, it remains unclear which one is essential. Furthermore, knowledge distilling with two large models is complicated in implementation.

III. NEURAL SOURCE-FILTER MODELS

We propose a framework of neural waveform modeling that is fast in generation and straightforward in implementation. As Figure 1 illustrates, our proposed framework contains three modules: a condition module that processes input acoustic features, source module that produces a sine-based excitation signal given the F0, and neural filter module that converts the excitation into a waveform using dilated convolution (CONV)

and feedforward transformation. Rather than the MSE or the likelihood over waveform sampling points, the proposed framework uses spectral-domain distances for model training. Because our proposed framework explicitly uses a sine-based excitation signal, we refer to all of the models based on the framework as NSF models.

Our proposed NSF framework does not rely on AR or IAF approaches. An NSF model converts an excitation signal into an output waveform without sequential transformation. It can be simply trained using the stochastic gradient descent (SGD) method under a spectral domain criterion. Therefore, its time complexity is theoretically irrelevant to the waveform length, i.e., $\mathcal{O}(1)$ in both the model training and waveform generation stages. Neither does an NSF model use knowledge distilling, which makes it straightforward in training and implementation.

An NSF model can be implemented in various architectures. We gives details on the three specific NSF models:

1) The b-NSF model has a network structure partially similar to ClariNet and Parallel WaveNet;
2) The s-NSF model inherits b-NSF's structure but uses much simpler neural filter modules;
3) The hn-NSF model extends the s-NSF and explicitly generates the harmonic and noise components of a waveform.

Although the three NSF models differ in terms of network structure, they use the same spectral-domain training criterion. We therefore first explain the training criterion in Section III-A then the three NSF models in Sections III-B to III-D.

A. Training criterion based on spectral amplitude distances

A good metric $\mathcal{L}(\widehat{\boldsymbol{o}}_{1:T}, \boldsymbol{o}_{1:T}) \in \mathbb{R}_{\geq 0}$ should measure the distance between the perceptual quality of a generated waveform $\widehat{\boldsymbol{o}}_{1:T}$ and that of a natural waveform $\boldsymbol{o}_{1:T}$. Additionally, a gradient $\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{o}}_{1:T}}$ based on the metric should be easily calculated so that a model can be trained using the SGD method. For the NSF models, we use spectral amplitude distances calculated on the basis of short-time Fourier transform (STFT). Although spectral amplitude distance has been used in classical speech coding methods [30], [31], [32], we further compute multiple distances with different short-time analysis configurations, as illustrated in Figure 2.

1) Forward propagation: In the forward propagation, we follow the STFT convention and conduct framing and windowing on the waveforms, after which the spectrum of each frame is computed using the discrete Fourier transform (DFT). Let us use $\widehat{\boldsymbol{x}}^{(n)} = [\widehat{x}^{(n)}_1, \cdots, \widehat{x}^{(n)}_M]^\top \in \mathbb{R}^M$ to denote the n-th frame of the generated waveform $\widehat{\boldsymbol{o}}_{1:T}$, where M is the

---

[2]The original WaveGlow paper used eight V100 GPU cards for model training [17]. However, it did not report the amount of training time.

frame length, and $\widehat{x}_m^{(n)}$ is the value of a windowed waveform sample. We then use $\widehat{\boldsymbol{y}}^{(n)} = [\widehat{y}_1^{(n)}, \cdots, \widehat{y}_K^{(n)}]^\top \in \mathbb{C}^K$ to denote the spectrum of $\widehat{\boldsymbol{x}}^{(n)}$ calculated using K-point DFT. We similarly define $\boldsymbol{x}^{(n)}$ and $\boldsymbol{y}^{(n)}$ for the natural waveform $\boldsymbol{o}_{1:T}$. Suppose $\widehat{\boldsymbol{o}}_{1:T}$ and $\boldsymbol{o}_{1:T}$ have been sliced into N frames. A log spectral amplitude distance $\mathcal{L}$ can be calculated as

$$\mathcal{L} = \frac{1}{2NK} \sum_{n=1}^{N} \sum_{k=1}^{K} \Big[\log \frac{\mathrm{Re}(y_k^{(n)})^2 + \mathrm{Im}(y_k^{(n)})^2 + \eta}{\mathrm{Re}(\widehat{y}_k^{(n)})^2 + \mathrm{Im}(\widehat{y}_k^{(n)})^2 + \eta} \Big]^2, \quad (5)$$

where $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ denote the real and imaginary parts of a complex number, respectively, and $\eta = 1e-5$ is a constant number to ensure numerical stability.

2) Backward propagation: For model training, we need to calculate the gradient vector $\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{o}}_{1:T}} \in \mathbb{R}^T$ and propagate it back to the model. Although $\mathcal{L}$ is calculated given complex-valued spectra, it turns out that $\frac{\partial \mathcal{L}_s}{\partial \widehat{\boldsymbol{o}}_{1:T}}$ can be computed efficiently.

First, because $\widehat{\boldsymbol{y}}^{(n)} = [\widehat{y}_1^{(n)}, \cdots, \widehat{y}_K^{(n)}]^\top \in \mathbb{C}^K$ is the spectrum of $\widehat{\boldsymbol{x}}^{(n)} = [\widehat{x}_1^{(n)}, \cdots, \widehat{x}_M^{(n)}]^\top \in \mathbb{R}^M$, we know that

$$\mathrm{Re}(\widehat{y}_k^{(n)}) = \sum_{m=1}^{M} \widehat{x}_m^{(n)} \cos \big(\frac{2\pi}{K}(k-1)(m-1) \big), \quad (6)$$

$$\mathrm{Im}(\widehat{y}_k^{(n)}) = - \sum_{m=1}^{M} \widehat{x}_m^{(n)} \sin \big(\frac{2\pi}{K}(k-1)(m-1) \big), \quad (7)$$

where $k \in \{1, \cdots, K\}^3$. Because $\mathrm{Re}(\widehat{y}_k^{(n)})$, $\mathrm{Im}(\widehat{y}_k^{(n)})$, $\frac{\partial \mathcal{L}}{\partial \mathrm{Re}(\widehat{y}_k^{(n)})}$, and $\frac{\partial \mathcal{L}}{\partial \mathrm{Im}(\widehat{y}_k^{(n)})}$ are real-valued numbers, we can compute the gradient by using the chain rule:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}} &= \sum_{k=1}^{K} \Big[\frac{\partial \mathcal{L}}{\partial \mathrm{Re}(\widehat{y}_k^{(n)})} \frac{\partial \mathrm{Re}(\widehat{y}_k^{(n)})}{\partial \widehat{x}_m^{(n)}} + \frac{\partial \mathcal{L}}{\partial \mathrm{Im}(\widehat{y}_k^{(n)})} \frac{\partial \mathrm{Im}(\widehat{y}_k^{(n)})}{\partial \widehat{x}_m^{(n)}} \Big] \\
&= \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial \mathrm{Re}(\widehat{y}_k^{(n)})} \cos(\frac{2\pi}{K}(k-1)(m-1)) - \\
&\quad \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial \mathrm{Im}(\widehat{y}_k^{(n)})} \sin(\frac{2\pi}{K}(k-1)(m-1)).
\end{aligned}$$
$$(8)$$

As long as we can compute $\frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}}$ for each m and n, the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_t}$ for $t \in \{1, \cdots, T\}$ can be easily accumulated from $\frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}}$ given the relationship between $\boldsymbol{o}_t$ and each $\widehat{x}_m^{(n)}$ that is determined by the framing and windowing operations.

Although we can use matrix multiplication to implement Equation (8), a more efficient method is to use the inverse-DFT. Suppose a complex-valued signal $\boldsymbol{g}^{(n)} \in \mathbb{C}^K$ can be composed for the n-th frame as $\boldsymbol{g}^{(n)} = [g_1^{(n)}, \cdots, g_K^{(n)}]$, where

$$g_k^{(n)} = \frac{\partial \mathcal{L}}{\partial \mathrm{Re}(\widehat{y}_k^{(n)})} + j \frac{\partial \mathcal{L}}{\partial \mathrm{Im}(\widehat{y}_k^{(n)})} \in \mathbb{C}. \quad (9)$$

---

[3] The summation in Equations (6)-(7) should be $\sum_{m=1}^{K}$, but the zero-padded part $\sum_{m=M+1}^{K} 0 \cos(\frac{2\pi}{K}(k-1)(m-1))$ can be safely ignored. Although we can avoid zero-padding by setting $K = M$, in practice, K is usually the power of 2 to take advantage of the fast Fourier transform (FFT). A waveform frame of length M can be zero-padded to length $K = 2^{\lceil \log_2 M \rceil}$ or longer to increase the frequency resolution.
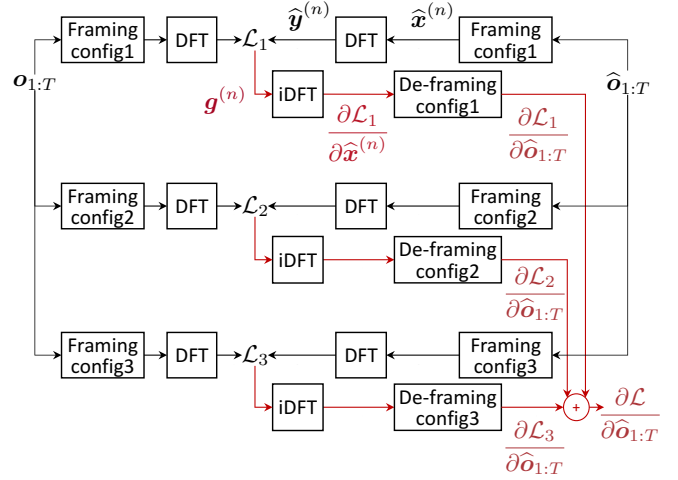

Fig. 2. Illustration of calculating three spectral distances for forward (black) and backward (red) propagation. DFT denotes discrete Fourier transform. Vectors $\widehat{\boldsymbol{x}}^{(n)}$, $\widehat{\boldsymbol{y}}^{(n)}$, and $\widehat{\boldsymbol{g}}^{(n)}$ denote windowed waveform, spectrum, and composed gradient vector for n-th frame.

If $\boldsymbol{g}^{(n)}$ is conjugate symmetric, i.e.,

$$\mathrm{Re}(g_k^{(n)}) = \mathrm{Re}(g_{(K+2-k)}^{(n)}), \quad k \in [2, \frac{K}{2}] \quad (10)$$

$$\mathrm{Im}(g_k^{(n)}) = \begin{cases} -\mathrm{Im}(g_{(K+2-k)}^{(n)}), & k \in [2, \frac{K}{2}] \\ 0, & k = \{1, \frac{K}{2}+1\}, \end{cases} \quad (11)$$

the inverse-DFT of $\boldsymbol{g}^{(n)}$ will be a real-valued signal $\boldsymbol{b}^{(n)} = [b_1^{(n)}, \cdots, b_K^{(n)}] \in \mathbb{R}^K$, where

$$\begin{aligned}
b_m^{(n)} &= \sum_{k=1}^{K} g_k^{(n)} e^{j \frac{2\pi}{K}(k-1)(m-1)} \\
&= \sum_{k=1}^{K} \mathrm{Re}(g_k) \cos(\frac{2\pi}{K}(k-1)(m-1)) - \\
&\quad \sum_{k=1}^{K} \mathrm{Im}(g_k) \sin(\frac{2\pi}{K}(k-1)(m-1)).
\end{aligned} \quad (12)$$

By combining Equations (9) and (12), we can see that $b_m^{(n)}$ is equal to $\frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}}$ in Equation (8). In other words, if $\boldsymbol{g}^{(n)}$ is conjugate symmetric, we can calculate $\frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}}$ by constructing $\boldsymbol{g}^{(n)}$ and taking its inverse-DFT. It can be shown that $\boldsymbol{g}^{(n)}$ is conjugate symmetric when $\mathcal{L}$ is defined as the log spectral amplitude distance in Equation (5). Other common distances such as the Kullback-Leibler divergence (KLD) between spectra [33], [34] are also applicable.

Note that, if $\widehat{\boldsymbol{x}}^{(n)}$ is zero-padded from length M to length K before DFT, the inverse-DFT of $\boldsymbol{g}^{(n)}$ will contain gradients w.r.t the zero-padded part. In such a case, we can simply assign $\frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}} \leftarrow b_m^{(n)}, \forall m \in \{1, \cdots, M\}$ and ignore $\{b_{M+1}^{(n)}, \cdots, b_K^{(n)}\}$ that correspond to the zero-padded part.

3) Multi-resolution spectral amplitude distance: The explanation on forward and backforward propagation shows that $\mathcal{L} \in \mathbb{R}$ and $\frac{\partial \mathcal{L}_s}{\partial \widehat{\boldsymbol{o}}_{1:T}} \in \mathbb{R}^T$ can be computed no matter how we configure the window length, frame shift, and DFT bins, i.e., the values of N, M, K. It is thus straightforward to merge

multiple distances $\{\mathcal{L}_1, \cdots, \mathcal{L}_S\}$ with different windowing and framing configurations. In such a case, the ultimate distance can be defined as $\mathcal{L} = \mathcal{L}_1 + \cdots + \mathcal{L}_S$. Accordingly, the gradients can be merged as $\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{o}}_{1:T}} = \frac{\partial \mathcal{L}_1}{\partial \hat{\boldsymbol{o}}_{1:T}} + \cdots + \frac{\partial \mathcal{L}_S}{\partial \hat{\boldsymbol{o}}_{1:T}}$.

Using multiple spectral distances is expected to help the model learn the spectral details of natural waveforms in different spatial and temporal resolutions. We used three distances in this study, which are explained in Section IV-B.

4) Remark on spectral amplitude distance: The short-time spectral amplitude distances may be more appropriate than the waveform MSE in Equation (1) for the NSF models. For a single speech frame, it is assumed with an NSF model using a spectral amplitude distance that the spectral amplitude vector $\boldsymbol{z} = \boldsymbol{y} \odot \boldsymbol{y}^* \in \mathbb{R}^K$ follows a multivariate log-normal distribution with a diagonal covariance matrix, where $\boldsymbol{y} = \text{DFT}(\boldsymbol{x}) \in \mathbb{C}^K$ is the K-point DFT of a waveform frame $\boldsymbol{x}$ and $\odot$ denotes element-wise multiplication. Although we cannot derive an analytical form of $p(\boldsymbol{x})$ from $p(\boldsymbol{z})$, we can at least infer that the distribution of $\boldsymbol{x}$ or the original waveform $\boldsymbol{o}_{1:T}$ assumed with the model is at least not an isotropic Gaussian distribution. An NSF model with a spectral amplitude distance can potentially model the waveform temporal correlation within an analysis window.

Using the spectral amplitude distance is reasonable also because the perception of speech sounds are affected by the spectral acoustic cues such as formants and their transition [35], [36], [37]. Although the spectral amplitude distance ignores other acoustic cues, such as phase [38] and timing [39], we only considered the spectral amplitude distance in this study because we have not found a phase or timing distance that is differentiable and effective.

B. Baseline NSF model

We now give the details on the b-NSF model. As Figure 3 illustrates, the b-NSF model uses three modules to convert an input acoustic feature sequence $\boldsymbol{c}_{1:B}$ of length B into a speech waveform $\hat{\boldsymbol{o}}_{1:T}$ of length T: a source module that generates an excitation signal $\boldsymbol{e}_{1:T}$, a filter module that transforms $\boldsymbol{e}_{1:T}$ into an output waveform, and a condition module that processes $\boldsymbol{c}_{1:B}$ for the source and filter modules. The model is trained using the spectral distance explained in the previous section.

1) Condition module: The input of the condition module is $\boldsymbol{c}_{1:B} = \{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_B\}$, where $\boldsymbol{c}_b = [\tilde{f}_b, \boldsymbol{s}_b^\top]^\top$ includes the F0 value $\tilde{f}_b$ and the spectral features $\boldsymbol{s}_b$ for the b-th frame. The F0 sub-sequence $\{\tilde{f}_1, \cdots, \tilde{f}_B\}$ is upsampled to $\boldsymbol{f}_{1:T}$ by replicating each $\tilde{f}_b$ for $\lceil T/B \rceil$ times, after which $\boldsymbol{f}_{1:T}$ is fed to the source module. The spectral features are processed by a bi-directional recurrent layer with long-short-term memory (LSTM) units [40] and a 1-dimensional CONV layer with a window size of 3. The processed spectral features are then concatenated with the F0 and upsampled as $\tilde{\boldsymbol{c}}_{1:T}$. The layer size of the Bi-LSTM and CONV layers is 64 and 63, respectively. The dimension of $\tilde{\boldsymbol{c}}_t$ is 64. Note that there is no golden network structure for the condition module. We used the structure in Figure 3 because it has been used in our

WaveNet-vocoder [41] [4].

2) Source module: Given the F0, the source module constructs an excitation signal on the basis of sine waveforms and random noise. In voiced segments, the excitation signal is a mixture of sine waveforms whose frequency values are determined by F0 and its harmonics. In unvoiced regions, the excitation signal is a sequence of Gaussian noise.

The input F0 sequence is $\boldsymbol{f}_{1:T}$, where $f_t \in \mathbb{R}_{\geq 0}$ is the F0 value of the t-th time step, and $f_t > 0$ and $f_t = 0$ denote being voiced and unvoiced, respectively. A sine waveform $\boldsymbol{e}_{1:T}^{<0>}$ with the fundamental frequency can be generated as

$$ e_t^{<0>} = \begin{cases} \alpha \sin(\sum_{k=1}^{t} 2\pi \frac{f_k}{N_s} + \phi) + n_t, & \text{if } f_t > 0 \\ \frac{\alpha}{3\sigma} n_t, & \text{if } f_t = 0 \end{cases}, \quad (13) $$

where $n_t \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise, $\phi \in [-\pi, \pi]$ is a random initial phase, and N_s is the waveform sampling rate. The hyper-parameter α adjusts the amplitude of source waveforms, while σ is the standard deviation of the Gaussian noise[5]. We set $\sigma = 0.003$ and $\alpha = 0.1$ in this study. Equation (13) treats f_t as an instantaneous frequency [42]. Thus, the phase of the $\boldsymbol{e}_{1:T}^{<0>}$ becomes continuous even if f_t changes. Figure 5 plots an example $\boldsymbol{e}_{1:T}^{<0>}$ and the corresponding $\boldsymbol{f}_{1:T}$.

The source module also generates harmonic overtones. For the h-th harmonic overtone, which corresponds to the $(h+1)$-th harmonic frequency, an excitation signal $\boldsymbol{e}_{1:T}^{<h>}$ is generated from Equation (13) with $(h+1)f_t$. The source module then uses a trainable feedforward (FF) layer with a tanh activation function to combine $\boldsymbol{e}_{1:T}^{<0>}$ and $\boldsymbol{e}_{1:T}^{<h>}$ into the final excitation signal $\boldsymbol{e}_{1:T} = \{e_1, \cdots, e_T\}$, where $e_t \in \mathbb{R}, \forall t \in \{1, \cdots, T\}$. This combination can be written as

$$ \boldsymbol{e}_{1:T} = \tanh(\sum_{h=0}^{H} w_h \boldsymbol{e}_{1:T}^{<h>} + w_b), \quad (14) $$

where $\{w_0, \cdots w_H, w_b\}$ are the FF layer's weights, and H is the total number of overtones.

The value of H is not critical to the model's performance because the model can re-create higher harmonic tones, as the experiments discussed in Section IV-D demonstrated. We set $H = 7$ based on a tentative rule $(H+1) * f_{max} < N_s/4$, where $N_s = 16$ kHz is the sampling rate, and $f_{max} \approx 500$ Hz is the largest F0 value observed in our data corpus. We used $N_s/4$ as the upper-bound so that there is at least four sampling points in each period of the sine waveform.

3) Neural filter module: The filter module of the b-NSF model transforms the excitation $\boldsymbol{e}_{1:T}$ into an output waveform $\hat{\boldsymbol{o}}_{1:T}$ by using five baseline dilated-CONV filter blocks. The structure of a baseline filter block is illustrated in Figure 4.

Suppose the input to the block is $\boldsymbol{v}_{1:T}^{\text{in}}$, where $v_t^{\text{in}} \in \mathbb{R}$, $\forall t \in \{1, \cdots, T\}$. This input $\boldsymbol{v}_{1:T}^{\text{in}}$ is expanded in dimension through an FF layer as $\tanh(\boldsymbol{w} v_t^{\text{in}} + \boldsymbol{b}) \in \mathbb{R}^{64}, \forall t \in \{1, \cdots, T\}$,

---

[4]Our previous NSF model [20] had no "concatenate" block in the condition module. In this study we added the "concatenate" block so that our NSF models use exactly the same condition module as our WaveNet-vocoder.

[5]In our previous NSF paper [20], we used $\frac{1}{3\sigma} n_t$ for the noise excitation. In this study, we used $\frac{\alpha}{3\sigma} n_t$ so that the amplitude of the noise in unvoiced segments is comparable to that of the sine waveforms in voiced segments.
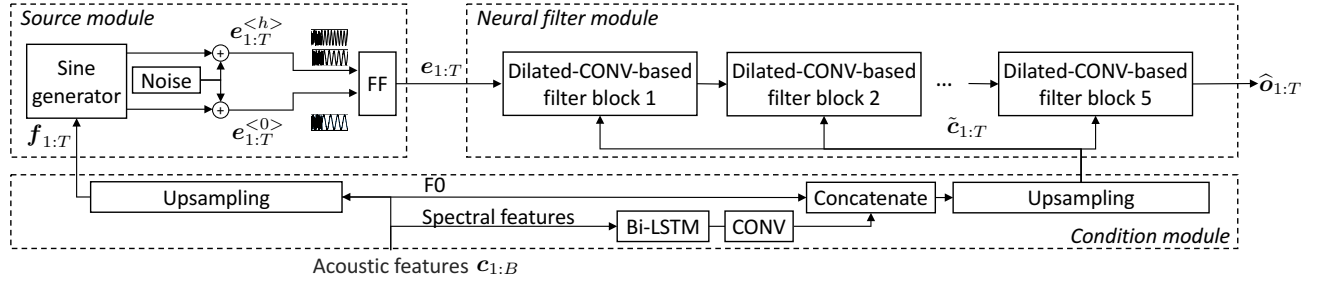
Fig. 3. Structure of baseline NSF (b-NSF) model. B and T denote lengths of input feature sequence and output waveform, respectively. FF, CONV, and Bi-LSTM denote feedforward, convolutional, and bi-directional recurrent layers, respectively. Structure of dilated-CONV filter block is plotted in Figure 4.
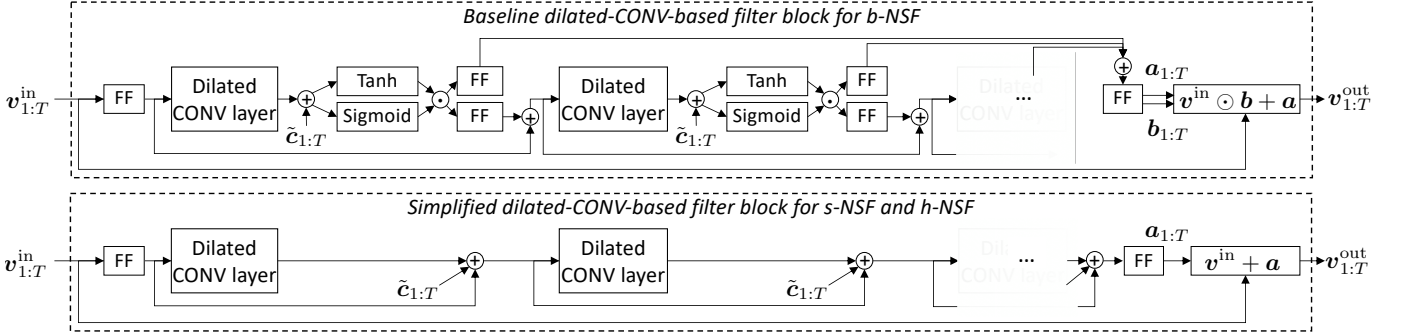

Fig. 4. Structure of dilated-CONV-based filter blocks for b-NSF model (top) and simplified NSF (s-NSF) model (bottom). $\boldsymbol{v}_{1:T}^{\text{in}}$ and $\boldsymbol{v}_{1:T}^{\text{out}}$ denote input and output of one filter block, respectively. $\odot$ denotes element-wise product. Every filter block contains 10 dilated-convolution layers, and every CONV and FF layer use tanh activation function.
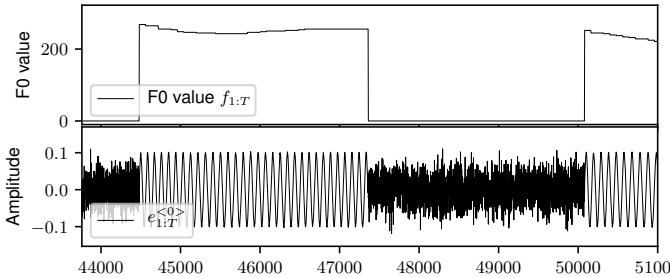

Fig. 5. Example of F0 sequence $\boldsymbol{f}_{1:T}$ and fundamental component $\boldsymbol{e}_{1:T}^{<0>}$

where $w \in \mathbb{R}^{64 \times 1}$ is the transformation matrix and $\boldsymbol{b} \in \mathbb{R}^{64}$ is the bias. The expanded signal is then processed by a dilated-CONV layer, summed with the condition feature $\tilde{\boldsymbol{c}}_{1:T}$, processed by the gated activation unit based on tanh and sigmoid [12], and transformed by the FF layers. This procedure is repeated ten times within this filter block, and the dilation size of the dilated convolution layer in the k-th stage is set to $2^{\text{mod}(k\text{-}1, \, 10)}$. The outputs from the ten stages are summed and transformed into two signals $\boldsymbol{a}_{1:T}$ and $\boldsymbol{b}_{1:T}$. After that, $\boldsymbol{v}_{1:T}^{\text{in}}$ is converted into an output signal $\boldsymbol{v}_{1:T}^{\text{out}}$ by $\boldsymbol{v}_{1:T}^{\text{out}} = \boldsymbol{v}_{1:T}^{\text{in}} \odot \boldsymbol{b}_{1:T} + \boldsymbol{a}_{1:T}$, where $\odot$ denotes element-wise multiplication. The output $\boldsymbol{v}_{1:T}^{\text{out}}$ is further processed in the following filter block, and the output of the last filter block is the generated waveform $\widehat{\boldsymbol{o}}_{1:T}$.

Our implementation used a kernel size of 3 for the dilated-CONV layers, which is supposed to be necessary for non-AR waveform models [18]. Both the input and output feature vectors of the dilated-CONV layer have 64 dimensions. Accordingly, the residual connection that connects two adjacent dilated-CONV layers also has 64 dimensions. The feature vectors to the FF layer that produces $\boldsymbol{a}_{1:T}$ and $\boldsymbol{b}_{1:T}$ have 128 dimensions, i.e., skip-connection of 128 dimensions. The $\boldsymbol{b}_{1:T}$ is parameterized as $\boldsymbol{b}_{1:T} = \exp(\tilde{\boldsymbol{b}}_{1:T})$ to be positive [19].

The baseline dilated-CONV filter block is similar to the student models in ClariNet and Parallel WaveNet because all use the stack of so-called "dilated residual blocks" in AR WaveNet [12]. However, because the b-NSF model does not use knowledge distilling, it is unnecessary to compute the distribution of the signal during forward propagation as ClariNet and Parallel WaveNet do. Neither is it necessary to make the filter blocks invertible as IAF does. Accordingly, the dilated convolution layers can be non-causal, even though we used causal ones to keep configurations of our NSF models consistent with our WaveNet-vocoder in the experiments.

C. Simplified NSF model

The network structure of the b-NSF model, especially the filter module, was designed on the basis of our experience with implementing WaveNet. However, we found that the filter module can be simplified, as shown in Figure 4. Such a filter block keeps only the dilated-CONV layers, skip-connections, and FF layers for dimension change. The output of a filter block is the sum of a residual signal $\boldsymbol{a}_{1:T}$ and the input signal $\boldsymbol{v}_{1:T}^{\text{in}}$. Using the sum $\boldsymbol{v}_{1:T}^{\text{out}} = \boldsymbol{v}_{1:T}^{\text{in}} + \boldsymbol{a}_{1:T}$ rather than the affine transformation $\boldsymbol{v}_{1:T}^{\text{out}} = \boldsymbol{v}_{1:T}^{\text{in}} \odot \boldsymbol{b}_{1:T} + \boldsymbol{a}_{1:T}$ was motivated by the result of our ablation test on the b-NSF model (Section IV-C: see the results of N1). Note that each dilated-CONV layer uses the tanh activation function.

On the basis of the simplified filter block, we constructed the s-NSF model. Compared with the b-NSF model, the s-NSF
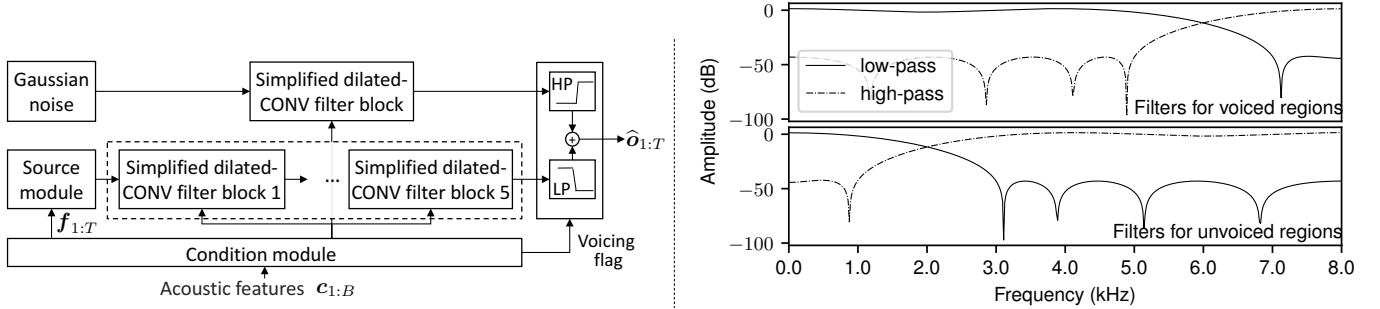
Fig. 6. Diagram of harmonic-plus-noise NSF (hn-NSF) model and frequency responses of low-pass (LP) and high-pass (HP) filters for voiced and unvoiced regions.

model has the same network structure except the simplified filter blocks. Accordingly, the s-NSF model has fewer parameters and a faster generation speed. The cascade of simplified filter blocks also turns the filter module into a deep residual network [43].

D. Harmonic-plus-noise NSF model

Although the b-NSF and s-NSF models perform well in many cases, we found that both may produce low-quality unvoiced sounds or sometimes silence for fricative consonants. Analysis on the neural filter modules of a well-trained model suggests that the hidden features for the voiced sounds have a waveform-like temporal structure, while those for the unvoiced sounds are noisy. We thus hypothesize that voiced and unvoiced sounds may require different non-linear transformations in filter modules.

A better strategy may be to generate a periodic and noise component separately and merge them with different ratios into voiced and unvoiced sounds, an idea similar to the harmonic-plus-noise model [44], [45], [46]. Following the literature, we also refer to the periodic component as the harmonic component.

As an implementation, we constructed the hn-NSF model, which is illustrated in Figure 6. While the hn-NSF model uses the same modules as the s-NSF model to generate a waveform for the harmonic component, it uses only a noise excitation and simplified filter block for the noise component. The noise and harmonic components are filtered by a high-pass and low-pass digital filter, respectively, and are summed as the output waveform. Digital filters were also used to merge the noise and periodic signals in the classical speech modeling methods [47], [48]. The difference is that the hn-NSF model uses filters to directly merge the waveforms rather than the source signals.

Because the harmonic component is usually abundant in the voiced sounds, while the noise component dominates the unvoiced sounds, we use two pairs of low- and high-pass filters in the hn-NSF model, one pair for voiced sounds and the other for unvoiced sounds. We implemented the filters as equiripple finite impulse response (FIR) filters and computed their coefficients by using the Parks-McClellan algorithm [49]. The parameters of the filters are listed in Table II, and the frequency responses of the filters are plotted on the right side of Figure 6. Note that the order of the FIR filter is determined

TABLE II
PARAMETERS OF EQUIRIPPLE LOW- AND HIGH-PASS FIR FILTERS FOR
hn-NSF. PASSBAND RIPPLE AMPLITUDE IS LESS THAN 5 dB, AND
STOPBAND ATTENUATION IS -40 dB. FILTER COEFFICIENTS ARE
CALCULATED USING PARKS-MCCLELLAN ALGORITHM [49].

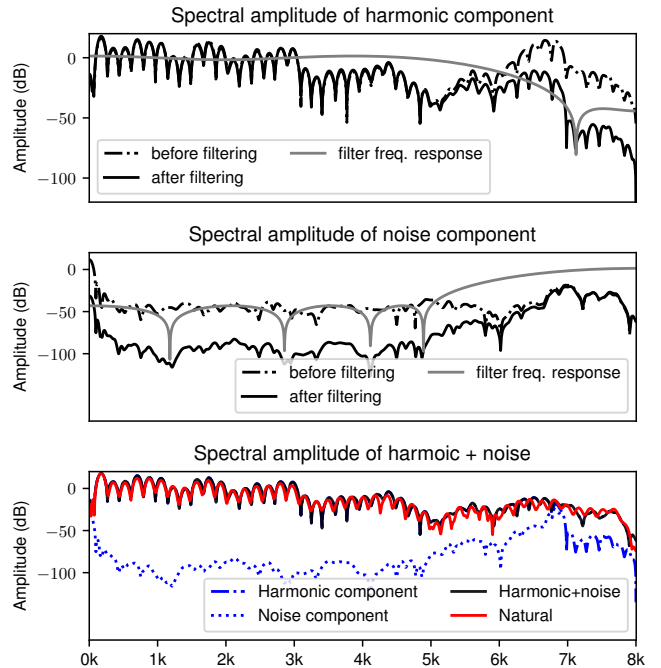|	Low-pass FIR filter		High-pass FIR filter	
	passband	stopband	passband	stopband
Voiced sound	$0 - 5$ kHz	$7 - 8$ kHz	$7 - 8$ kHz	$0 - 5$ kHz
Unvoiced sound	$0 - 1$ kHz	$3 - 8$ kHz	$3 - 8$ kHz	$0 - 1$ kHz

Fig. 7. Spectral amplitude and filter frequency response of one voiced speech frame generated by `hn-NSF` (Section IV-D).

by the Parks-McClellan algorithm. For the filters specified in Figure 6, the filter order is around 10. After the filter coefficients are calculated using the algorithm, they are stored and fixed in the model. The voicing flag for selecting the filter can be easily extracted from the input F0 sequence.

Although the low- and high-pass FIR filters are fixed, the neural filter blocks can learn to compensate and fine-tune the energy of the generated signals in certain frequency bands. Figure 7 plots the spectral amplitudes of the harmonic

and noise components of one voiced frame generated by the hn-NSF model. For the harmonic component, although the passband of the low-pass filter is only up to 5kHz, the neural filter module generates a harmonic component with a high energy above between 5 and 7 kHz, which compensates for the attenuation of the low-pass filter. As the last row of Figure 7 shows, the harmonic component dominates the generated waveform frame from 0 to 7 kHz, and its spectral amplitude is similar to that of natural speech.

IV. EXPERIMENTS

Following the explanation on our NSF models, we now discuss the experiments. After describing the corpus and data configuration in Section IV-A, we describe an ablation test conducted on the b-NSF model in Section IV-C. We then compare the three NSF models with WaveNet in Section IV-D. In Section IV-E, we investigate the controllability of the input F0 on the NSF models, and in Section IV-F, we examine the internal behaviors of the NSF models.

A. Corpus and data configuration

Our experiments used a data set of neural-style reading speech by a Japanese female speaker (F009), which is part of the XIMERA speech corpus [50]. This data set was recorded at a sampling rate of 48 kHz and segmented into 30,016 utterances. The total duration is around 50 hours.

We prepared three training subsets for the experiments on waveform modeling: the first subset contained 9,000 randomly selected utterances (15 hours), the second included 3,000 utterances (5 hours) randomly selected from the first subset, and the third contained 1,000 utterances (1.6 hours) randomly selected from the second subset. The first and third subsets were used to evaluate the performance of the NSF models and WaveNet in Section IV-D. The second subset was used in the ablation test on the NSF models in Section IV-C. We also prepared a validation set with 500 utterances and a test set with another 480 utterances. All utterances were downsampled to 16 kHz for waveform model training.

The acoustic features were extracted from the natural waveforms with a frame shift of 5 ms (200 Hz). The F0 values were extracted by an ensemble of pitch trackers [51]. Two types of spectral features were prepared: Mel-generalized cepstral coefficients (MGCs) [52] of order 60 extracted using the WORLD vocoder [53] and Mel-spectrogram of order 80. We used the F0 and either the MGCs or the Mel-spectrogram as the input features to the waveform models.

To evaluate the waveform models in SPSS TTS systems, we also trained acoustic models to predict acoustic features from linguistic features. To train acoustic models, we extracted linguistic features from text, including quin-phone identity, phrase accent type, and other structural information [54]. These features were force-aligned with the acoustic features by using hidden Markov models.

B. Model configurations

Four waveform models were evaluated in the experiments: our three NSF models b-NSF, s-NSF, and hn-NSF and AR

WaveNet (WaveNet). We chose WaveNet as the benchmark because of its excellent performance reported in both the original paper [12] and our previous study [13][55]. We did not include IAF-flow-based models due to their high demand on training time and GPU resources. Neither did we consider Parallel WaveNet or ClariNet due to the lack of authentic implementation.

The network configuration of b-NSF was described in Section III-B. It used five baseline dilated-CONV filter blocks, and each block contained ten dilated CONV and other hidden layers, as illustrated in Figure 4. The dilation size of the k layer was 2^{k-1}. s-NSF was the same as b-NSF except that each baseline dilated-CONV filter block was replaced with a simplified version. hn-NSF used the same network as s-NSF for the harmonic component and a single simplified block for the noise component. All three NSF models used the spectral amplitude distance $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3$, and the configuration of each $\mathcal{L}_*$ is listed in Table III. Note that $\mathcal{L}_1$ used the same frame shift and frame length as those for extracting the acoustic features from the waveforms. $\mathcal{L}_2$ and $\mathcal{L}_3$ were decided so that one has a higher temporal resolution while the other has a higher frequency resolution than $\mathcal{L}_1$. The configurations in Table III may be inappropriate for a different corpus, and a good configuration may be found through trial and error.

Our WaveNet used the same network structure as that in our previous study [13]. It used 40 dilated-CONV layers in total, where the k-th one had a dilation size of $2^{\mathrm{modulo}(k-1,10)}$. Between two adjacent CONV layers, WaveNet used a network structure similar to that in the baseline filter block of b-NSF. However, each dilated-CONV layer used a filter size of 2 and 128 output channels, and the condition feature $\tilde{c}_t$ was transformed by an additional FF layer into 128 dimensions before being summed with the dilated-CONV layer's output. The skip-connection had 256 dimensions, and the output vectors of the 40 dilated-CONV stages were propagated by the skip-connection and summed together. The summed vector was transformed into a 1024-dimensional activation vector to a softmax function, which calculates the categorical distribution for the 10-bit quantized μ-law waveform value. The condition module of WaveNet was the same as the NSF models. During generation, WaveNet selected the most probable waveform value from the distribution for 25% of the voiced time steps. Otherwise, it randomly drew a sample as the output[6].

All the neural waveform models were trained on a single-GPU card (Nvidia Tesla P100) using the Adam optimizer [56]

---

[6]This new strategy slightly improved the aperiodicity of voiced sounds, compared with the WaveNet in our previous study, which selected the most probable waveform value at all voiced steps [13]

with a learning rate= 0.0003, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The training process was terminated when the error on the validation set continually increases for five epochs. The batch size was 1, and each utterance was truncated into segments of at most 3 seconds to fit the GPU memory.

All the neural waveform models were implemented using a modified CURRENNT toolkit [57]. Using the same toolkit allows us to fairly compare the models since they use the same set of low-level CUDA/THUST functionalities [58]. Note that our WaveNet implementation is sufficiently good as the benchmark. As another study on the same corpus demonstrated [55], our WaveNet implementation can generate speech waveforms that are similar to the original natural waveforms in terms of perceived quality, given natural acoustic features [7]. The code, scripts, and samples are publicly available at https://nii-yamagishilab.github.io/samples-nsf/nsf-v2.html.

For the acoustic models that predict acoustic features from the linguistic features, we used shallow and deep neural AR models [9], [5] to generate the MGCs and F0, respectively. We trained another deep AR model to generate the Mel-spectrogram. The recipes for training these three acoustic models were the same as those in another of our previous study [54]. The number of training utterances for acoustic models was around 28,000 (47 hours of data). The acoustic feature sequences were generated given the duration force-aligned on the test set.

C. Ablation test on b-NSF

Although we claimed that an NSF model can be implemented in varied network architectures, some of the components may be essential to model performance. This ablation test was conducted to identify those essential components.

We used b-NSF as the reference model and prepared a few variants, as listed in Table IV. All the models including b-NSF were trained using the MGCs, F0, and waveforms from the 5-hr. training set. The trained models then generated waveforms given the natural acoustic features in the test set, and these generated waveforms were evaluated in a subjective evaluation test. In one evaluation round, an evaluator listened to one speech waveform on one screen, rated the speech quality on a 1-to-5 mean-opinion-score (MOS) scale, and repeated the process for multiple screens. The waveforms in one evaluation round were for the same text and were played in a random order. All the waveforms were converted to 16-bit PCM format in advance.

A total of 245 paid Japanese native speakers participated in the test, and 1444 valid evaluation rounds were conducted. The results are plotted in Figure 8, and the difference between b-NSF and the other models was statistically significant ($p < 0.01$), as two-sided Mann-Whitney tests demonstrated. First, a comparison made among b-NSF, L1, L2, and L3 showed that using spectral amplitude distances with different windowing and framing configurations is essential to the model's performance. The waveforms generated from L1, L2, and L3 were perceptually worse because of a pulse-train-like

[7]Unlike naive open-source implementations, our WaveNet-vocoder avoids redundant CONV operations as the so-called Fast-WaveNet did [59]

TABLE IV
MODELS FOR ABLATION TEST (SECTION IV-C)

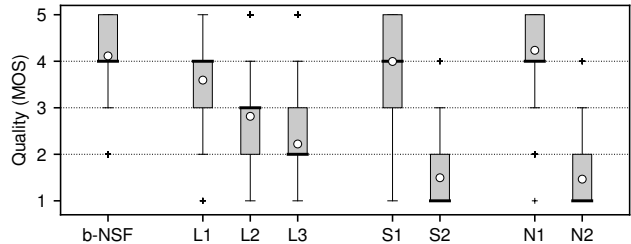| Model	Description	
b-NSF	trained on 5-hr. training set	
L1	b-NSF without using $\mathcal{L}_3$	(i.e., $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$)
L2	b-NSF without using $\mathcal{L}_2$	(i.e., $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_3$)
L3	b-NSF without using $\mathcal{L}_2$ or $\mathcal{L}_3$	(i.e., $\mathcal{L} = \mathcal{L}_1$)
S1	b-NSF without harmonics overtones (i.e., H=0 in Equation (14))	
S2	b-NSF using noise as source signal (i.e., $e_{1:T} = \frac{\alpha}{3\sigma} n_{1:T}$)	
N1	b-NSF with $\boldsymbol{b}_{1:T} = 1$ in filter blocks	
N2	b-NSF with $\boldsymbol{b}_{1:T} = 0$ in filter blocks	

Fig. 8. Mean opinion scores (MOSs) of synthetic samples from b-NSF and its variants given natural acoustic features. White dots denote mean MOSs.

sound in both unvoiced and voiced sounds. This type of artifact could be easily observed in the unvoiced segments plotted in Figure 9. We hypothesize that using spectral distances with different temporal-spatial resolutions could mitigate the pulse-train-like artifacts in the spectrogram.

By comparing b-NSF, S1, and S2, we found that the sine-based excitation is essential to the NSF models. In the case of S2, the generated waveforms were intelligible but unnatural because the perceived pitch was unstable. As Figure 9 shows, the waveform generated from S2 lacked the periodic structure that should be observed in voiced sounds. With a sine-based excitation, the b-NSF model may have a better starting point to generate waveforms with a periodic structure. This hypothesis is supported by the results of the investigation discussed in Section IV-F.

Interestingly, N1 outperformed b-NSF even though it used a simpler transformation in the filter blocks. In comparison, the waveforms generated from N2 were unnatural because they lacked the stable periodic structures in voiced segments. One possible reason is that the skip-connection enables the sine excitation to be propagated to the later filter blocks without being attenuated by the non-linear transformations.

In summary, the results of the ablation test suggest that both the sine excitation and multi-resolution spectral distances are crucial to NSF models. It is also important to keep the skip-connections inside the filter modules.

D. Comparison between WaveNet and NSF models

1) Quality of generated waveforms: This experiment compared b-NSF, s-NSF, hn-NSF, and WaveNet under four training conditions. Each model was trained using either the 15-hr. or 1.6-hr. training set and conditioned on F0 and either Mel-spectrogram or MGCs. In the testing stage, each model generated speech waveforms given natural acoustic features
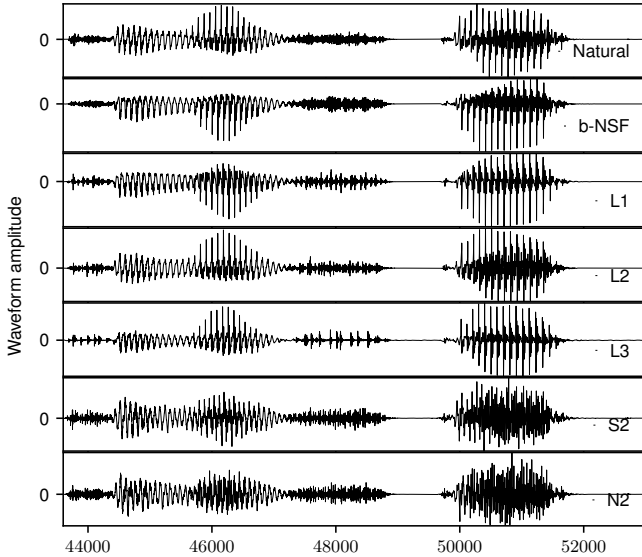
Fig. 9. Natural and generated waveforms from NSF models in ablation test (Section IV-C).

TABLE V
MODEL COMPARISON IN TERMS OF TRAINING SPEED, GENERATION SPEED, AND NUMBER OF MODEL PARAMETERS. TRAINING SPEED WAS MEASURED ON 15-HR AND 1.6-HR TRAINING DATA SETS. TRAINING AND GENERATION WERE CONDUCTED ON SINGLE NVIDIA P100 GPU CARD.

	Training speed (hr/epoch)		No. of generated waveforms points in 1-s GPU time		No. of model parameters
Model	1.6-hr set	15-hr set	mem-save	normal	
WaveNet	0.46	3.01	-	0.19 k	2.96e+6
b-NSF	0.37	3.40	20 k	227 k	1.83e+6
s-NSF	0.25	1.81	78 k	343 k	1.07e+6
hn-NSF	0.27	1.92	71 k	335 k	1.20e+6

or generated features produced by the acoustic models. Accordingly, each model was trained and tested under eight conditions. The generated speech waveforms were evaluated in a subjective evaluation test, which was organized in the same manner as that mentioned in Section IV-C.

The results are plotted in Figure 10. Among the three NSF models, hn-NSF performed better than or comparably well with the other two versions. Interestingly, the Mel-spectrogram-based s-NSF performed poorly when it was trained using the 15-hr. set. One reason was that s-NSF produced low-quality unvoiced sounds. For example, the unvoiced segments generated by s-NSF had a very small amplitude, as shown in Figure 11. b-NSF performed well when it was trained using MGCs and F0 from the 15-hr. training set, but its performance was worse than hn-NSF and b-NSF when the amount of training data was less than 2 hours. One hypothesis is that b-NSF requires more training data since it has more parameters, as Table V shows.

A comparison made between hn-NSF and WaveNet shows that hn-NSF was comparable to WaveNet in terms of the generated speech quality. Specifically, in the TTS application, hn-NSF trained on 15 hours of Mel-spectrogram data slightly outperformed WaveNet and b-NSF trained on 15 hours of MGC data, which were the best performing models

in our previous study [20].

2) Training and generation speed: After the MOS test, we compared the waveform training and generation speed of the experimental models. Although the theoretical time complexity is described in Table 1, we measured the actual speed of each model in training and generation stages.

Table V lists the time cost to train the experimental models for one epoch. The training time cost on the 15-hr training set was larger than that on the 1.6-hr set because the number of training utterance increased. Nevertheless, the results indicate that the NSF models are comparable to WaveNet in terms of training speed. This is expected because all three NSF models and WaveNet require no sequential transformation of waveforms during model training.

For waveform generation, our implementation of the NSF models has normal and memory-saving generation modes. The normal mode allocates all of the required GPU memory once but cannot generate very long waveforms due to the limited memory space on a single GPU card. The memory-saving mode supports the generation of long waveforms because it releases and allocates the GPU memory layer by layer. However, these memory operations cost processing time.

We evaluated the NSF models in both modes by using a test subset with 80 test utterances, each of which was around 5 seconds. As the results in Table V indicate, the NSF models were much faster than WaveNet even in the memory-save mode. Note that WaveNet requires no repeated memory operation or memory-save mode. It is slow because of the AR-generation process. Compared with b-NSF, s-NSF was faster in generation because of its simplified network structure. Although hn-NSF slightly lagged behind s-NSF in terms of generation speed, it outperformed b-NSF.

In summary, the results of the MOS and speed tests indicate that hn-NSF performed no worse than WaveNet in terms of the quality of the generated waveforms. Furthermore, hn-NSF outperformed WaveNet by a large margin in term of waveform generation speed[8].

E. Consistency between input F0 and F0 in generated waveforms

The ablation test discussed in Section IV-C demonstrated that sine excitation with the input F0 is essential to the NSF models. In this experiment, we investigated the consistency between the input F0 and the F0 of the waveforms generated from the NSF models, especially when the F0 input to the source module is not identical to the F0 in the input Mel-spectrogram.

This experiment was conducted on hn-NSF and WaveNet, which were trained using the natural Mel-spectrogram and F0 from the 15-hr. training set. The WaveNet was included as a reference model. Before generating the waveforms, we used the deep AR F0 model to randomly generate three F0 contours for each of the test set utterances [5]. In this random generation mode, the three F0 contours for the same test utterance were slightly different from each other [5]. Let **F0r1**, **F0r2**, and

---

[8]The generation speed in the memory-save mode may be further increased if the GPU memory allocation can be accelerated.
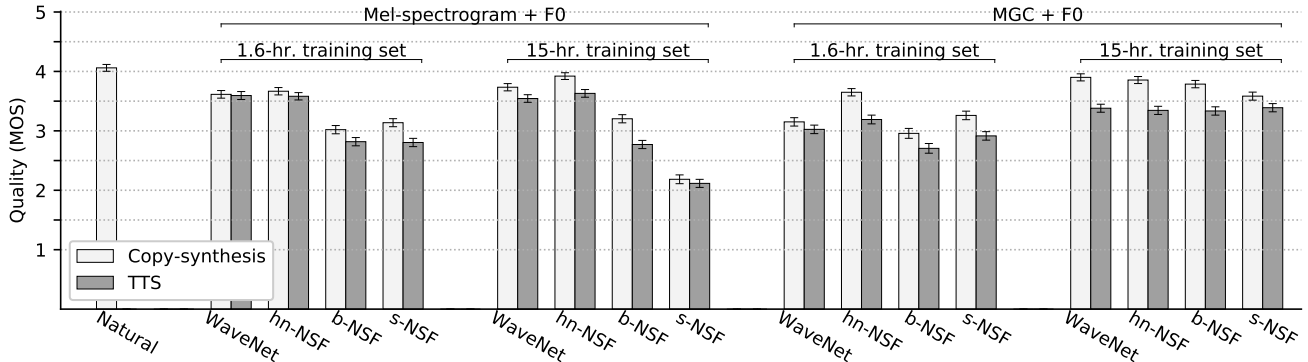
Fig. 10. MOSs of experimental systems under different training and test conditions. Error bars at confidence level of 95% are plotted.

(a) Using natural acoustic features for waveform generation

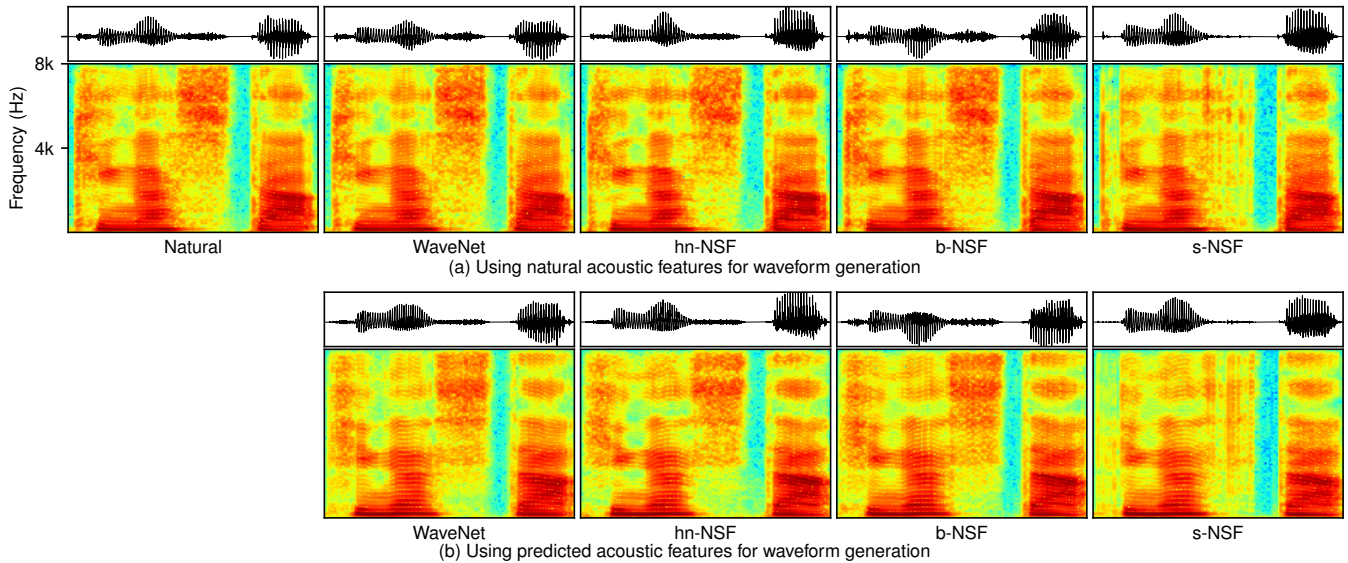(b) Using predicted acoustic features for waveform generation

Fig. 11. Natural and generated waveforms and their spectrograms. Models were trained using 15 hours of natural Mel-spectrogram and F0.
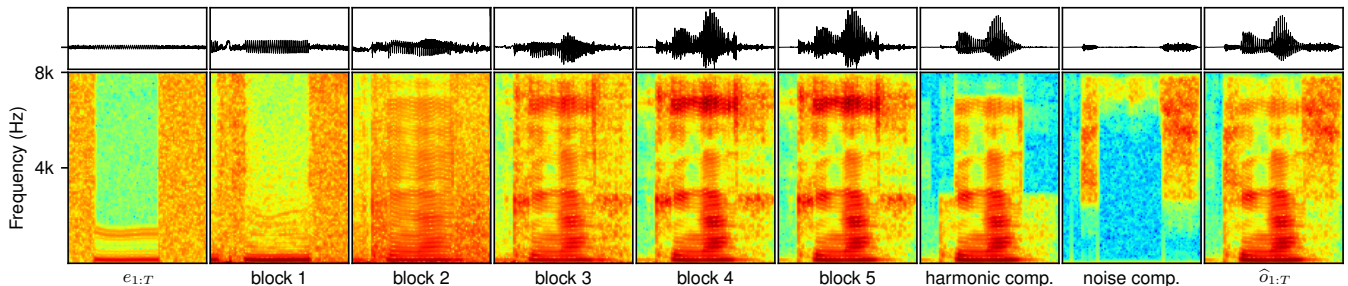

Fig. 12. Input, output, and hidden signals from hn-NSF given natural Mel-spectrogram and F0. From left to right: input excitation $e_{1:T}$, outputs from 5 neural filter blocks for harmonic waveform component, filtered harmonic and noise waveform components, and output waveform $\widehat{o}_{1:T}$.

F0r3 denote the three sets of F0 contours. We then used the three F0 sets and the generated Mel-spectrogram as the input to hn-NSF and WaveNet, which resulted in six sets of generated waveforms.

For each of the six sets, we calculated the correlation between the input F0 and the F0 extracted from the generated waveforms. For reference, we also extracted the F0 from the input Mel-spectrograms using a neural-network-based method [60]. The results listed in Table VI indicate that the F0 contours of the waveforms generated from the NSF models were highly consistent with the F0 input to the source module. However, the waveforms generated from WaveNet correlated with the F0 information buried in the input Mel-spectrogram.

The results indicate that we can easily control the F0 of the generated waveforms from the NSF models through directly manipulating the input F0. In contrast, it is less straightforward in the case of WaveNet because we have to manipulate the F0 contained in the input Mel-spectrogram.

F. Investigation of hidden features of hn-NSF

We argued in Section III-C that the simplified neural filter module in the s-NSF and hn-NSF models is similar to a deep residual network. It is thus interesting to look inside the neural filter module. From hn-NSF trained on 15 hours of Mel-spectrogram and F0, we generated one test utterance given natural condition data and extracted the one-dimensional

TABLE VI
F0 CORRELATION BETWEEN F0 INPUT TO WAVEFORM MODELS AND F0
EXTRACTED FROM WAVEFORMS GENERATED BY `hn-NSF` AND `WaveNet`.
NOTE THAT `hn-NSF` AND `WaveNet` USED MEL-SPECTROGRAM AND F0
IN **F0R1**, **F0R2**, OR **F0R3**.

	hn-NSF			WaveNet		
	F0r1	**F0r2**	**F0r3**	**F0r1**	**F0r2**	**F0r3**
F0r1	**0.992**	0.930	0.921	0.917	0.916	0.914
F0r2	0.926	**0.986**	0.919	0.918	0.919	0.917
F0r3	0.926	0.929	**0.988**	0.922	0.921	0.920
F0 in Mel-spect.	0.901	0.907	0.900	**0.975**	**0.971**	**0.973**

output of each filter block (i.e., the $q_{1:T}$ in Figure 4) in the sub-network to generate the harmonic waveform component. We also extracted the sine-based excitation $e_{1:T}$, filtered harmonic and noise waveform components, and final output waveform $\widehat{o}_{1:T}$. These signals and their spectrograms are plotted in Figure 12.

We can observe that the dilated-CONV filter blocks morphed the sine excitation into the waveform. The spectrogram of the sine excitation had no formant structure but only the fundamental frequency and harmonics. From blocks 1 to 5, the spectrogram of the signal was gradually enriched with the formant structure. The results also suggest that `hn-NSF` kept the F0 of the sine excitation in the output waveform. This explains why the F0 of the waveform generated from `hn-NSF` was highly consistent with the frequency of the sine excitation, or the input F0, the experiments of Section IV-E.

Similar results were observed when we analyzed `s-NSF` and `b-NSF`. For `b-NSF`, the results are consistent with the ablation test where we found that `b-NSF` without the skip-connections in the filter module performed poorly (`N2` in Section IV-C). The skip-connections make the filter module a deep residual network based on which the excitation signal can be gradually transformed into the output waveform.

These results also indicate how the sine excitation eases the task of waveform modeling because the neural filter modules do not need to reproduce the periodic structure that evokes the perception of F0 in voiced sounds. Without the sine excitation, it may be difficult for the neural filter modules to generate the periodic structure, which explains the poor performance of the b-NSF model without sine excitation (`S2` in Section IV-C).

V. CONCLUSION

We proposed a framework called "neural source-filter modeling" for the waveform models in TTS systems. A model implemented in this framework, which is called an "NSF model", can convert input acoustic features into a high-quality speech waveform. Compared with other neural waveform models such as WaveNet, an NSF model does not use an AR network structure and avoids the slow sequential waveform generation process. Neither does an NSF model use flow-based approaches nor knowledge distilling. Instead, an NSF model uses three modules that can be easily implemented: a source module that produces a sine-based excitation signal, filter module that transforms the excitation into an output waveform, and condition module that processes the input features for

the source and filter modules. Such an NSF model can be efficiently trained using a merged spectral amplitude distance. Even though this distance is calculated using multiple short time analysis configurations, it can be efficiently implemented on the basis of STFT. Therefore, the proposed NSF framework allows a neural waveform model to be built and trained straightforwardly.

Experimental results indicated that the specific hn-NSF model, which uses separate modules to model the harmonic and noise components of waveforms, performed comparably well to our WaveNet on a large single-speaker Japanese speech corpus. Furthermore, this NSF model can generate speech waveforms at a much faster speed. Another advantage of this NSF model is that the F0 input to the source module allows easy control on the pitch of the generated waveform.

In this primary study, we mainly described the NSF framework in detail and compared several NSF models with our verified WaveNet implementation. To further understand the NSF models' performance, we need a thorough comparison between NSF models and other types of neural waveform models on multi-speaker corpora. We leave this task for future work because of the time required to implement and train other models.

REFERENCES

[1] P. Taylor, *Text-to-Speech Synthesis*. Cambridge University Press, 2009.

[2] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura, "Speech synthesis based on hidden Markov models," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1234–1252, 2013.

[3] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, pp. 1039–1064, 2009.

[4] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," in *Proc. Interspeech*, 2015, pp. 3330–3334.

[5] X. Wang, S. Takaki, and J. Yamagishi, "Autoregressive neural F0 model for statistical parametric speech synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1406–1419, 2018.

[6] M. S. Ribeiro, O. Watts, and J. Yamagishi12, "Parallel and cascaded deep neural networks for text-to-speech synthesis," in *Proc. SSW*, 2016, pp. 100–105.

[7] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. ICASSP*, 2013, pp. 7962–7966.

[8] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *Proc. Interspeech*, 2014, pp. 1964–1968.

[9] X. Wang, S. Takaki, and J. Yamagishi, "An autoregressive recurrent mixture density network for parametric speech synthesis," in *Proc. ICASSP*, 2017, pp. 4895–4899.

[10] G. E. Henter, S. Ronanki, O. Watts, M. Wester, Z. Wu, and S. King, "Robust TTS duration modelling using DNNs," in *Proc. ICASSP*, 2016, pp. 5130–5134.

[11] B. Chen, T. Bian, and K. Yu, "Discrete duration model for speech synthesis," in *Proc. Interspeech*, 2017, pp. 789–793.

[12] A. van den 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.

[13] X. Wang, J. Lorenzo-Trueba, S. Takaki, L. Juvela, and J. Yamagishi, "A comparison of recent waveform generation and acoustic modeling methods for neural-network-based speech synthesis," in *Proc. ICASSP*, 2018, pp. 4804–4808.

[14] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," in *Proc. ICLR*, 2017.

[15] Y. Ai, H.-C. Wu, and Z.-H. Ling, "SampleRNN-based neural vocoder for statistical parametric speech synthesis," in *Proc. ICASSP*. IEEE, 2018, pp. 5659–5663.

[16] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *Proc. NIPS*, 2016, pp. 4743–4751.

[17] R. Prenger, R. Valle, and B. Catanzaro, "WaveGlow: A flow-based generative network for speech synthesis," in *Proc. ICASSP*, 2019, pp. 3617–3621.

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

[19] W. Ping, K. Peng, and J. Chen, "ClariNet: Parallel wave generation in end-to-end text-to-speech," in *Proc. ICLP*, 2019.

[20] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter-based waveform model for statistical parametric speech synthesis," in *Proc. ICASSP*, 2019, pp. 5916–5920.

[21] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[22] L. Juvela, V. Tsiaras, B. Bollepalli, M. Airaksinen, J. Yamagishi, and P. Alku, "Speaker-independent raw waveform model for glottal excitation," in *Proc. Interspeech 2018*, 2018, pp. 2012–2016.

[23] J.-M. Valin and J. Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *Proc. ICASSP*, 2019, pp. 5891–5895.

[24] M.-J. Hwang, F. Soong, F. Xie, X. Wang, and H.-G. Kang, "LP-WaveNet: Linear prediction-based WaveNet speech synthesis," *arXiv preprint arXiv:1811.11913*, 2018.

[25] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *Proc. ICML*, vol. 80, 10–15 Jul 2018, pp. 2410–2419.

[26] Z. Jin, A. Finkelstein, G. J. Mysore, and J. Lu, "FFTNet: A real-time speaker-dependent neural vocoder," in *Proc. ICASSP*. IEEE, 2018, pp. 2251–2255.

[27] T. Okamoto, K. Tachibana, T. Toda, Y. Shiga, and H. Kawai, "An investigation of subband WaveNet vocoder covering entire audible frequency range with limited acoustic features," in *Proc. ICASSP*. IEEE, 2018, pp. 5654–5658.

[28] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. ICML*, 2015, pp. 1530–1538.

[29] T. Okamoto, T. Toda, Y. Shiga, and H. Kawai, "Investigations of real-time neural vocoders with fundamental frequency and Mel-cepstra," in *Proc. ASJ spring meeting*, 2019, p. (in Japanese).

[30] D. Griffin and J. Lim, "A new model-based speech analysis/synthesis system," in *Proc. ICASSP*, vol. 10. IEEE, 1985, pp. 513–516.

[31] D. W. Griffin and J. S. Lim, "Multiband excitation vocoder," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 8, pp. 1223–1235, Aug 1988.

[32] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. ASSP*, vol. 32, no. 2, pp. 236–243, 1984.

[33] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. NIPS*, 2001, pp. 556–562.

[34] S. Takaki, H. Kameoka, and J. Yamagishi, "Direct modeling of frequency spectra and waveform generation based on phase recovery for DNN-based speech synthesis," in *Proc. Interspeech*, 2017, pp. 1128–1132.

[35] D. B. Fry, A. S. Abramson, P. D. Eimas, and A. M. Liberman, "The identification and discrimination of synthetic vowels," *Language and speech*, vol. 5, no. 4, pp. 171–189, 1962.

[36] A. M. Liberman, F. S. Cooper, D. P. Shankweiler, and M. Studdert-Kennedy, "Perception of the speech code," *Psychological review*, vol. 74, no. 6, p. 431, 1967.

[37] W. Strange, "Evolving theories of vowel perception," *The Journal of the Acoustical Society of America*, vol. 85, no. 5, pp. 2081–2087, 1989.

[38] I. Saratxaga, I. Hernaez, M. Pucher, E. Navas, and I. Sainz, "Perceptual importance of the phase related information in speech," in *Proc. Interspeech*, 2012.

[39] J. M. Hillenbrand and T. M. Nearey, "Identification of resynthesized/hvd/utterances: Effects of formant contour," *The Journal of the Acoustical Society of America*, vol. 105, no. 6, pp. 3509–3523, 1999.

[40] A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks," Ph.D. dissertation, Technische Universität München, 2008.

[41] X. Wang, S. Takaki, and J. Yamagishi, "Investigation of WaveNet for text-to-speech synthesis," SIG Technical Reports, Tech. Rep. 6, feb 2018.

[42] J. R. Carson and T. C. Fry, "Variable frequency electric circuit theory with application to the theory of frequency-modulation," *Bell System Technical Journal*, vol. 16, no. 4, pp. 513–540, 1937.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[44] A. J. Abrantes, J. S. Marques, and I. M. Trancoso, "Hybrid sinusoidal modeling of speech without voicing decision," in *Proc. Eurospeech*, 1991, pp. 231–234.

[45] J. Laroche, Y. Stylianou, and E. Moulines, "HNS: Speech modification based on a harmonic+ noise model," in *Proc. ICASSP*, vol. 2. IEEE, 1993, pp. 550–553.

[46] Y. Stylianou, "Harmonic plus noise models for speech, combined with statistical methods, for speech and speaker modification," Ph.D. dissertation, Ecole Nationale Superieure des Telecommunications, 1996.

[47] J. Makhoul, R. Viswanathan, R. Schwartz, and A. Huggins, "A mixed-source model for speech compression and synthesis," *The Journal of the Acoustical Society of America*, vol. 64, no. 6, pp. 1577–1581, 1978.

[48] A. V. McCree and T. P. Barnwell, "A mixed excitation LPC vocoder model for low bit rate speech coding," *IEEE Transactions on Speech and audio Processing*, vol. 3, no. 4, pp. 242–250, 1995.

[49] T. Parks and J. McClellan, "Chebyshev approximation for nonrecursive digital filters with linear phase," *IEEE Transactions on Circuit Theory*, vol. 19, no. 2, pp. 189–194, 1972.

[50] H. Kawai, T. Toda, J. Ni, M. Tsuzaki, and K. Tokuda, "XIMERA: A new TTS from ATR based on corpus-based technologies," in *Proc. SSW5*, 2004, pp. 179–184.

[51] L. Juvela, X. Wang, S. Takaki, S. Kim, M. Airaksinen, and J. Yamagishi, "The NII speech synthesis entry for Blizzard Challenge 2016," in *Proc. Blizzard Challenge Workshop*, 2016.

[52] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai, "Mel-generalized cepstral analysis a unified approach," in *Proc. ICSLP*, 1994, pp. 1043–1046.

[53] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[54] H.-T. Luong, X. Wang, J. Yamagishi, and N. Nishizawa, "Investigating accuracy of pitch-accent annotations in neural-network-based speech synthesis and denoising effects," in *Proc. Interspeech*, 2018, pp. 37–41.

[55] Y. Yasuda, X. Wang, S. Takaki, and J. Yamagishi, "Investigation of enhanced Tacotron text-to-speech synthesis systems with self-attention for pitch accent language," in *Proc. ICASSP*. IEEE, 2019, pp. 6905–6909.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014, p. unknown.

[57] F. Weninger, J. Bergmann, and B. Schuller, "Introducing CURRENT: The Munich open-source CUDA recurrent neural network toolkit," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 547–551, 2015.

[58] N. Bell and J. Hoberock, "Thrust: A productivity-oriented library for CUDA," in *GPU computing gems Jade edition*. Elsevier, 2011, pp. 359–371.

[59] T. L. Paine, P. Khorrami, S. Chang, Y. Zhang, P. Ramachandran, M. A. Hasegawa-Johnson, and T. S. Huang, "Fast WaveNet generation algorithm," *arXiv preprint arXiv:1611.09482*, 2016.

[60] L. Juvela, B. Bollepalli, X. Wang, H. Kameoka, M. Airaksinen, J. Yamagishi, and P. Alku, "Speech waveform synthesis from MFCC sequences with generative adversarial networks," in *Proc. ICASSP*. IEEE, 2018, pp. 5679–5683.
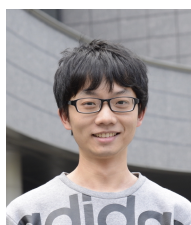
Xin Wang (S'16 - M'18) received his Ph.D. degree from Department of Informatics, SOKENDAI / National Institute of Informatics, Japan, 2018. He is now a project researcher at National Institute of Informatics, Japan. His research interests include statistical speech synthesis and machine learning.
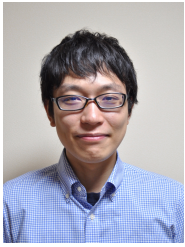
Shinji Takaki (M'16) received the B.E. degree in computer science, the M.E. and Ph.D. degrees in scientific and engineering simulation from Nagoya Institute of Technology, Nagoya, Japan in 2009, 2011, and 2014, respectively. From September 2013 to January 2014, he was a visiting researcher at University of Edinburgh. From April 2014, he was a project researcher at National Institute of Informatics. From April 2019, he is a project researcher at Nagoya Institute of Technology. He received the IPSJ Yamashita SIG Research Award, the ASJ Awaya Kiyoshi Award, in 2017 and 2019, respectively. His research interests include statistical machine learning and speech synthesis.

Junichi Yamagishi (SM'13) is a professor at National Institute of Informatics in Japan. He is also a senior research fellow in the Centre for Speech Technology Research (CSTR) at the University of Edinburgh, UK. He was awarded a Ph.D. by Tokyo Institute of Technology in 2006 for a thesis that pioneered speaker-adaptive speech synthesis and was awarded the Tejima Prize as the best Ph.D. thesis of Tokyo Institute of Technology in 2007. Since 2006, he has authored and co-authored over 200 refereed papers in international journals and conferences. He was awarded the Itakura Prize from the Acoustic Society of Japan, the Kiyasu Special Industrial Achievement Award from the Information Processing Society of Japan, and the Young Scientists' Prize from the Minister of Education, Science and Technology, the JSPS prize in 2010, 2013, 2014, and 2016, respectively.

He was one of organizers for special sessions on "Spoofing and Countermeasures for Automatic Speaker Verification" at Interspeech 2013, "ASVspoof evaluation" at Interspeech 2015, "Voice conversion challenge 2016" at Interspeech 2016, and "2nd ASVspoof evaluation" at Interspeech 2017. He has been a member of the Speech & Language Technical Committee (SLTC). He served as an Associate Editor of the IEEE/ACM Transactions on Audio, Speech and Language Processing and a Lead Guest Editor for the IEEE Journal of Selected Topics in Signal Processing (JSTSP) special issue on Spoofing and Countermeasures for Automatic Speaker Verification.