IPSJ SIG Technical Report

Vol.2018-MUS-118 No.6
Vol.2018-SLP-120 No.6
2018/2/20

Investigation of WaveNet for Text-to-Speech Synthesis

Xin Wang[1,a)] Shinji Takaki[1,b)] Junichi Yamagishi[1,2,c)]

Abstract: WaveNet is a type of neural network that can be used to model speech waveforms. It has been used in text-to-speech synthesis systems to convert acoustic or linguistic features into waveforms. Despite the description in recent literatures and open-source implementation, the mechanism of WaveNet is still somewhat obscure. This work explains the authors' WaveNet implementation. It also introduces a one-best generation method that could be an alternative to the random-sampling-based generation method. Based on the implementation, this work shows observations inside the network. Interesting findings include the manifold of quantized waveforms learned by WaveNet and the gradually decreased data variance in WaveNet blocks. These results may be helpful for further investigation on WaveNet.

Keywords: WaveNet, text-to-speech

1. Introduction

Text-to-speech synthesis (TTS) is a technique that converts a text string into a speech waveform. A classical TTS system derives linguistic features from the text and then generates a waveform. A recent breakthrough in TTS is credited to a type of neural network called WaveNet [1], which is able to model and generate quantized speech waveforms with natural-sounding quality. WaveNet can be used as the final block of a TTS system to convert acoustic features generated by preceding acoustic models into a waveform [2]. Alternatively, WaveNet can be directly used for waveform generation given linguistic features. This work refers to the WaveNet in the first case as *WaveNet-vocoder* while in the latter case as *WaveNet-backend*.

WaveNet is much larger than conventional neural networks for TTS. Although open-source codes of WaveNet are available, they are based on Tensorflow [3] or other high-level programming languages, because of which the details of computation may be veiled. Furthermore, those implementations may include specific structures without explaining the motivation. On the other hand, most of the recent literatures focus on the application of WaveNet. The inner side of WaveNet is not well explored.

This report summarizes the lessons that the authors learned from implementing and analyzing WaveNet. Section 2 will explain the implementation and the motivation for a few specific designs. This section also explains a generation method for WaveNet, which could be an alternative to the random-sampling-based approach. Section 3 will summarize the observations on WaveNet, including the manifold of quantized waveform and the decreased variance of the features in WaveNet skip-channels. This section also show the experiments on different generation methods for both WaveNet-vocoder and WaveNet-backend.

The implemented WaveNet is based on C++/CUDA. The source code and scripts to use it can be found from http://tonywangx.github.io.

2. WaveNet in details

2.1 Network structure

WaveNet is an autoregressive model. It defines the probability for observing a waveform $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$ conditioned on an acoustic or linguistic feature sequence $\boldsymbol{c}_{1:N} = \{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_N\}$ as

$$P(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:N}) = \prod_{t=1}^{T} P(\boldsymbol{o}_t|\boldsymbol{o}_{t-R:t-1}, \boldsymbol{c}_{1:N}). \tag{1}$$

Here, T is waveform length in terms of the number of sampling points; N is the number of speech frames; R denotes the size of receptive field, which will be explained later. Vector $\boldsymbol{o}_t$ is a one-hot vector that encodes the quantized waveform at time t. Its dimension is equal to the number of quantization level L. Vector $\boldsymbol{c}_n$ for WaveNet-based vocoder may include Mel-general cepstrum coefficients (MGC) [4] and F0 of the n-th frame. For WaveNet-based TTS backend, $\boldsymbol{c}_n$ may encode the F0 and linguistic features.

An essential task of any WaveNet implementation is to calculate $P(\boldsymbol{o}_t|\boldsymbol{o}_{t-R:t-1}, \boldsymbol{c}_{1:N}), \forall t \in \{1, \cdots, T\}$. Figure 1 shows the structure of the authors' WaveNet implementation, which includes waveform-embedding, dilated convolution backbone, conditional feature processing, and post-processing. Although Figure 1 only shows the computation at time t, in training stage $P(\boldsymbol{o}_t|\boldsymbol{o}_{t-R:t-1}, \boldsymbol{c}_{1:N})$ of all time steps can be computed simultaneously by using convolution and matrix operation. However, it is easier to explain WaveNet by focusing on a single time step.

2.1.1 Waveform-embedding

At time t, the waveform-embedding module takes $\boldsymbol{o}_{t-1}$ as input and transforms it as

$$\boldsymbol{e}_t = \boldsymbol{W}_{em}\boldsymbol{o}_{t-1}, \tag{2}$$

where $\boldsymbol{W}_{em} \in \mathbb{R}^{D_r \times L}$ and $\boldsymbol{e}_t, \boldsymbol{b}_{em} \in \mathbb{R}^{D_r}$. This module then feeds

1 National Institute of Informatics, Tokyo, Japan
2 The University of Edinburgh, Edinburgh, UK
a) wangxin@nii.ac.jp
b) takaki@nii.ac.jp
c) jyamagish@nii.ac.jp

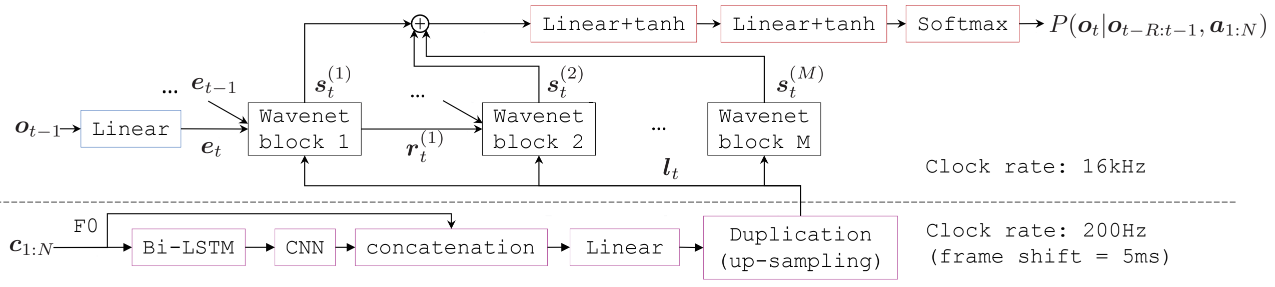

Fig. 1 General structure of implemented WaveNet. Blue block denotes waveform-embedding module; black blocks do dilated convolution and other operations; purple blocks process conditional features; red blocks do post-processing. Note that $^{(1)}, ^{(2)}, \cdots, ^{(M)}$ are the index of WaveNet block; ↘ denotes a feature vector from a previous time step, which is used by dilated convolution.

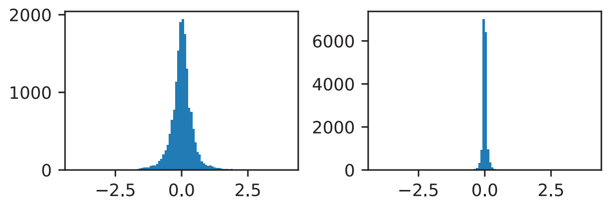

Fig. 2 Histogram on values of W_{eml} (left figure) and W_{emr} (right figure) from causal embedding layer.

e_t to Wavenet block 1. Because o_{t-1} is a one-hot vector, e_t is just one column of W_{em} selected by the hot-dimension of o_{t-1}. Since $L < D_r$, e_t is referred to as the embedding vector of o_{t-1}. Such an embedding vector is retrieved for every $t \in \{1, \cdots, T\}$.

In some other open-source codes of WaveNet [*1], the waveform-embedding module uses a so-called causal embedding layer, which defines $e_t = W_{eml} o_{t-1} + W_{emr} o_{t-2}$. However, it seems to be unnecessary to add o_{t-2} since information about o_{t-2} has been sent to the Wavenet block through e_{t-1}. In fact, when we trained a WaveNet with a causal embedding layer, we found that the value of elements in W_{emr} is close to zero as Figure 2 shows. It suggests that e_t focuses on o_{t-1}. Therefore, our implementation doesn't use the causal embedding layer.

2.1.2 WaveNet backbone

The backbone of WaveNet consists of multiple blocks, each of which does dilated convolution and other operations. Literatures usually use the terminology of image processing to explain those blocks, which makes it less clear for speech processing. Hence, we use the notation of linear transformation for explanation.

A block in the WaveNet backbone is referred to as a Wavenet block. We take the first block as an example and plot its structure in Figure 3. For brief notation, we drop the block index $^{(1)}$ for all the vectors inside the block. At time t, this block uses a dilated convolution layer to process the input vectors e_t and e_{t-1}, which are generated by the waveform-embedding module at time t and $t-1$. For a single time step, this dilated convolution is equivalent to linear transformation and summation [*2]:

$$h_t = W_{dcl} e_t + W_{dcr} e_{t-1} + b_{dc}, \quad (3)$$

[*1] For example https://github.com/ibab/tensorflow-wavenet

[*2] In the training stage where $e_{1:T} = \{e_1, \cdots, e_T\}$ have been calculated from the training waveform $o_{1:T}$, linear transformation and summation for every $t \in \{1, \cdots, T\}$ can be conducted at the same time in a parallel way. Because the same pair of transformation matrices $\{W_{dcl}, W_{dcr}\}$ is used for every t, all the computation can be implemented as a single 1D convolution on $e_{1:T}$.

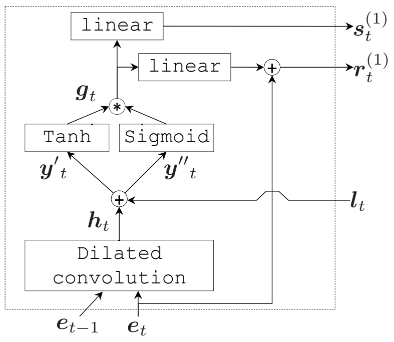

Fig. 3 Structure of wavenet block 1. Index $^{(1)}$ is dropped for all vectors inside the block.

where $W_{dcl}, W_{dcr} \in \mathbb{R}^{2D_r \times D_r}$, $h_t \in \mathbb{R}^{2D_r}$, and $b_{dc} \in \mathbb{R}^{2D_r}$. The output vector h_t can be added with $l_t \in \mathbb{R}^{2D_r}$ given by the conditional feature module:

$$y_t = h_t + l_t. \quad (4)$$

After splitting the vector y_t into two short vectors $y'_t \in \mathbb{R}^{D_r}$ and $y''_t \in \mathbb{R}^{D_r}$, where $y_t = [y'^{\top}_t, y''^{\top}_t]^{\top}$, activation functions can be used to derive another vector $g_t \in \mathbb{R}^{D_r}$ as

$$g_t = \tanh(y'_t) \odot \text{sigmoid}(y''_t). \quad (5)$$

Here, $\odot$ denotes the element-wise product. After that, g_t can be further transformed and then summed with e_t as

$$r_t^{(1)} = W_r g_t + b_r + e_t, \quad (6)$$

where $W_r \in \mathbb{R}^{D_r \times D_r}$ and $r_t^{(1)} \in \mathbb{R}^{D_r}$. This $r_t^{(1)}$ will go to the next wavenet block. Meanwhile, g_t can be transformed into another vector $s_t^{(1)}$:

$$s_t^{(1)} = W_s g_t + b_s, \quad (7)$$

where $W_s \in \mathbb{R}^{D_s \times D_r}$ and $b_s \in \mathbb{R}^{D_s}$. This $s_t^{(1)}$ is fed to the post-processing module.

Other wavenet blocks do similar operations except that the input vectors may be different. For example, the m-th block with a dilation size d should define Equation 3 as

$$h_t^{(m)} = W_{dcl}^{(m)} r_t^{(m-1)} + W_{dcr}^{(m)} r_{t-d}^{(m-1)} + b_{dc}^{(m)}, \quad (8)$$

where $r_t^{(m-1)}$ and $r_{t-d}^{(m-1)}$ are given by the $(m-1)$-th block at time t and $t-d$. Since $r_{t-d}^{(m-1)}$ may further depend on features extracted by the $(m-2)$-th block before $t-d$, the stack of WaveNet blocks can cover a long span of waveform. The length of the span is called the receptive field R.

The path that $\boldsymbol{r}_t^{(m)}$ goes through is called *residual channel* while that of $\boldsymbol{s}^{(m)_t}$ is called *skip channel*. Note that Equation 6 is different from the original WaveNet paper, which defines $\boldsymbol{r}_t^{(m)} = \boldsymbol{s}_t^{(m)} + \boldsymbol{e}_t$. In other words, the original WaveNet paper assumes that $\boldsymbol{s}_t^{(m)}$ and $\boldsymbol{r}_t^{(m)}$ have the same dimension. Normally, $\boldsymbol{s}_t^{(m)}$ should have a large dimension because it will be used by the post-processing block to calculate the probability. However, if $\boldsymbol{r}_t^{(m)}$ has the same dimension as $\boldsymbol{s}_t^{(m)}$, the whole network would be extremely large. Therefore, it is better to use separate transformations for $\boldsymbol{r}_t^{(m)}$ and $\boldsymbol{s}_t^{(m)}$, i.e., Equation 6 and 7. In this way, dimension of $\boldsymbol{r}_t^{(m)}$ and other vectors inside each block can be reduced. This structure is also used in other open-source WaveNet implementation and Deep voice [5].

2.1.3 Post-processing module

The post-processing module sums $\boldsymbol{s}_t^{(m)}$ over $m \in \{1, \cdots, M\}$. It then transforms the summed vector and calculates the output probability using a softmax layer. The dimension of $\boldsymbol{s}_t^{(m)}$, i.e., D_s, is called the size of skip-channel.

2.1.4 Conditional feature module

Suppose time step t locates in the n-th speech frame, then $\boldsymbol{l}_t$ in Equation 4 can be acquired by changing the dimension of $\boldsymbol{c}_n$. However, we found that it is better to generate $\boldsymbol{l}_t$ through a recurrent network. In this work, this module contains a bi-directional long-short-term-memory (LSTM) recurrent layer followed by a convolution layer with a window size of 3. Additionally, F0 is directly concatenated with the output vector of the convolution layer. After dimension change, the output vector is used as $\boldsymbol{l}_t$ in Equation 4. Note that vector of one frame is duplicated for all the time steps within that frame.

2.2 Generation method

In the generation stage, WaveNet predicts $\widehat{\boldsymbol{o}_t}$ given previously generated samples. According to the literature, $\widehat{\boldsymbol{o}_t}$ is randomly sampled from $P(\boldsymbol{o}_t|\widehat{\boldsymbol{o}}_{t-R:t-1}, \widehat{\boldsymbol{c}}_{1:N})$. However, we noticed that the generated waveform may sound hoarse in the case of WaveNet-vocoder. One reason could be that randomly sampled waveform may not preserve the harmonic structure of natural speech. After plotting the distribution inferred by WaveNet, we noticed that the distribution in voiced segments normally had a single peak. We think it better to trust the inferred distribution and directly pick the best point as output, i.e., $\widehat{\boldsymbol{o}_t} = \arg\max_{\boldsymbol{o}_t} P(\boldsymbol{o}_t|\widehat{\boldsymbol{o}}_{t-R:t-1}, \widehat{\boldsymbol{a}}_{1:N})$. This method is referred to as the *one-best generation* method. Note that samples in the unvoiced regions are still randomly drawn, and the voiced/unvoiced boundary are determined based on the unvoiced/voiced information in F0.

2.3 Memory and time consumption

Let's take the dilated convolution layer in WaveNet block 1 as example. Naively, a memory matrix of size $T \times D_r$ is required to store $\boldsymbol{e}_{1:T} = \{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_T\} \in \mathbb{R}^{T \times D_r}$. To compute $\boldsymbol{e}_{1:T}$ naively, Equation 3 should be conducted for every t. Similar time and memory consumption is required by all the hidden layers. Intuitively, the memory (or space) and time consumption of naive WaveNet implementation depends on the length of waveform T and the total number of layers.

Of course, the consumption can be reduced. In the training stage where the waveform $\boldsymbol{o}_{1:T}$ is known, embedding vectors $\boldsymbol{e}_{1:T}$ can be computed by launching Equation 2 for every t in a parallel way. Then, computation defined in Equation 3 can also be computed for $\forall t \in \{1, \cdots, T\}$ simultaneously, which is implemented by a single 1D convolution operation. Except the LSTM layer in conditional feature module, computation in any other layer can be conducted efficiently regardless of T. However, the memory consumption cannot be reduced as memory space is required to store the feature vectors for all $t \in \{1, \cdots, T\}$. In implementation, we simply split the training waveform to make sure the network can be trained by using a single GPU card. To model waveforms at the sampling rate of 16kHz, the maximum length of waveform is set to be no longer than 0.9375s, or $T \leq 15000$. Given this configuration, the memory consumption of the network used in experiment is around 12GB.

In the generation stage, computation must be conducted for every t sequentially. The maximum length of waveform should not be limited. Therefore, the time consumption is proportional to T and the total number of layers [*3]. However, the memory consumption can be reduced. First, it is unnecessary to allocate memory space of length T for hidden features that will be only used by time-independent operations. For example, we only allocate a memory of size D_r to store $\boldsymbol{y}_t$ at time t. After $\boldsymbol{y}_t$ is used, this memory space is ready to store $\boldsymbol{y}_{t+1}$ for the next time step. On the other hand, a memory space of size $d \times D_r$ must be allocated to store the input feature of a Wavenet block with dilation size d. At time t, this memory will store $\{\boldsymbol{r}_{t-d}^{(m-1)}, \cdots, \boldsymbol{r}_t^{(m-1)}\}$. Details of implementation can be found online [*4].

3. Experiments and results

3.1 Corpus and network configuration

This work used a Japanese corpus [6] of reading speech uttered by a female speaker. The duration is 50 hours, and 500 utteranaces were randomly selected as a validation set and another 500 were used as a test set. Linguistic features with a dimension of 389 were extracted by using OpenJTalk [7]. MGC and F0 were extracted from the 48 kHz waveforms by using WORLD [8] with a frame rate of 200 Hz (5 ms). The dimension for MGC was 60. For WaveNet-vocoder, the acoustic model for MGC was a shallow autoregressive (AR) network (SAR) [9]; F0 was modeled by a deep AR model [10]. Note that F0 was not interpolated but quantized.

Both WaveNet-vocoder and WaveNet-backend used the same structure. Waveforms were down-sampled to 16kHz, companded by using the μ-law, and then quantized into 10 bits per sample, i.e., $L = 1024$. The network had $M = 40$ blocks. The k-th block has a dilation size of $2^{\mathrm{mod}(k,10)}$, where $\mathrm{mod}(\cdot)$ is modulo operation.

[*3] However, this doesn't mean that the implementation is stupidly slow. In fact, the time consumption is equal to the so-called 'fast-wavenet'. We think it better to call the 'fast-wavenet' as 'normal-speed yet memory-friendly WaveNet' since it avoids the redundant computation in a 'slow WaveNet' while keeps the memory consumption at a low level. It should be noted that a 'normal-speed but memory-prohibitive WaveNet' can be built by simply caching $\boldsymbol{r}_t^{(m)}$ for every t and m. Compared with 'fast-wavenet', we use a plain buffer rather than a queue to cache $\boldsymbol{r}_t^{(m)}$

[*4] http://tonywangx.github.io/pdfs/CURRENNT_WAVENET.pdf

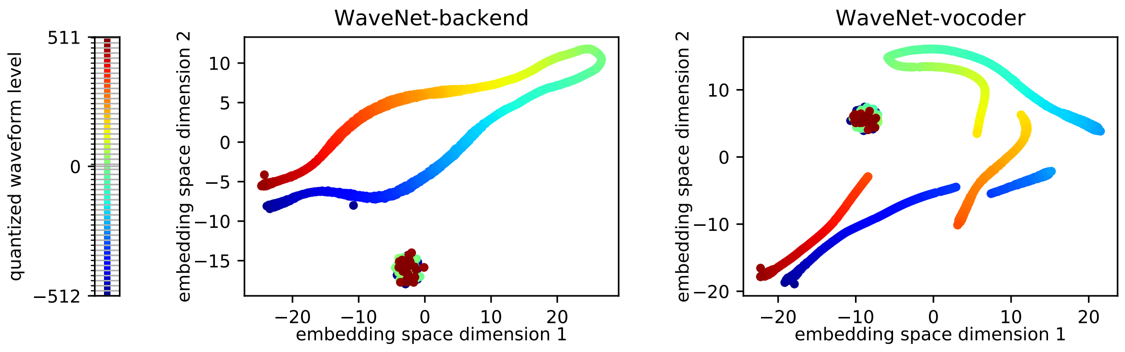

Fig. 4 Manifold of embedded vectors for quantized waveforms. Left figure shows the color assigned to each quantized waveform level.

The size of residual channel is $D_r = 64$; size of skip-channel is $D_s = 512$. In the conditional feature module, the layer size of the Bi-LSTM is 64; the output feature dimension of CNN is 60; the size of concatenated vector is 61. Natural acoustic features are used for training WaveNet-vocoder. The training recipe can be found online.

3.2 Manifold of quantized waveform

It is shown in Section 2.1.1 that $\boldsymbol{W}_{em} \in \mathbb{R}^{D_r \times L}$ in Equation 2 contains the embedding vectors for quantized waveform. Specifically, if we write $\boldsymbol{W}_{em} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_L]$, where $\boldsymbol{w}_l \in \mathbb{R}^{D_r}, l \in \{1, \cdots, L\}$, $\boldsymbol{w}_l$ is the embedding vector for the quantization level l. After training, we cast all the embedding $\boldsymbol{w}_l$ into a 2D space by using t-SNE [11]. The results are shown in Figure 4.

Figure 4 suggests that the embedded vectors lie on a low-dimensional manifold. Furthermore, the order of the embedded vectors on the manifold is consistent with the order of quantization levels, i.e., the indicated quantization level monotonically increase or decrease as we move from one end of the manifold to the other end. Although it has been argued that distance between two one-hot vectors cannot reflect the distance between the waveform values represented by the one-hot vectors, the learned manifold suggests that the network can infer the distance between one-hot vectors based on the distance between their embedding vectors. Interestingly, the manifold of WaveNet-vocoder is divided into several pieces. Reason for this result is still being investigated.

Note that vectors corresponding to the extremely large quantization level, which are colored by dark red and dark blue, formed one cluster. The reason may be that training data do not cover those waveform quantization levels. Therefore, those embedding vectors were not well updated during training. Vectors for quantization levels around 0 (green color) are also in that cluster. This may due to the fact that those levels may only corresponding to small noise in training data.

3.3 Data variance in residual channel

As Figure 3 shows that each WaveNet block contains a skip-connection, the sequence of WaveNet blocks formulates a residual structure [12]. It would be interesting to investigate the contribution of each block to the post-processing block. For this purpose, we collected $\boldsymbol{s}^{(m)}$ from all the blocks when WaveNet-vocoder generated waveforms on 10 randomly selected test utterances. Then, we flattened $\boldsymbol{s}^{(m)}$ and calculated the data variance over t. Figure 5 shows the data variance for each $m \in \{1, \cdots, M\}$.

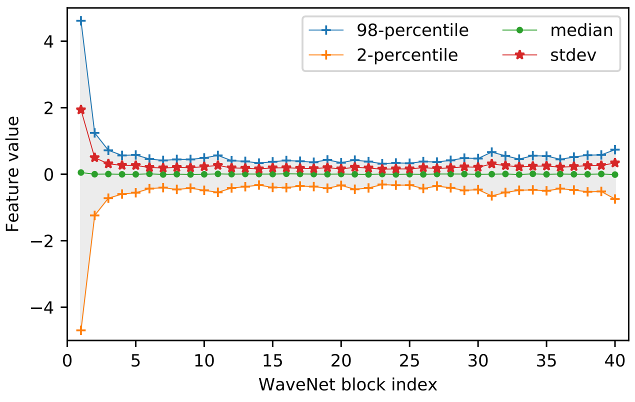

Fig. 5 Statistics on $\boldsymbol{s}_t^{(m)}$ over 10 test utterances for WaveNet-vocoder. The horizontal axis denotes index $m \in \{1, \cdots, M\}$.

The interesting observation is that the data variance decreases as m increases. This trend is somewhat consistent with the analysis on other residual networks [13]. It is argued that the initial block in a residual structure explains the general feature pattern while the following blocks finely modify the description given by the preceding block. According to the recent work on WaveNet [14], it is suggested that removing some Wavenet blocks may not severely degrade the overall performance. This may be possible according to the decreased data variance over $\boldsymbol{s}_t^{(m)}$. It is somewhat consistent with the finding on residual or highway network [13].

3.4 Comparing generation methods

For WaveNet-based vocoder, we found that the random-sampling-based method generated waveforms with hoarse quality, especially in the voiced segments. This may due to the fact that sampled waveform in a local segment may not convey the harmonic structure of a voiced sound. As Figure 6 suggests, the spectrum of a voiced sound generated by random-sampling (figure on the right-tpp corner) seems to be quite noisy around the harmonics over 4k Hz. Similarly, the figure on instantaneous frequency (figure on the right-bottom corner) doesn't show regular patterns that can be found in natural speech. The reason for this result may be explained by Figure 7, where the inferred distribution and the picked value are plotted for at a few time steps within that voiced sound. The second row shows that random

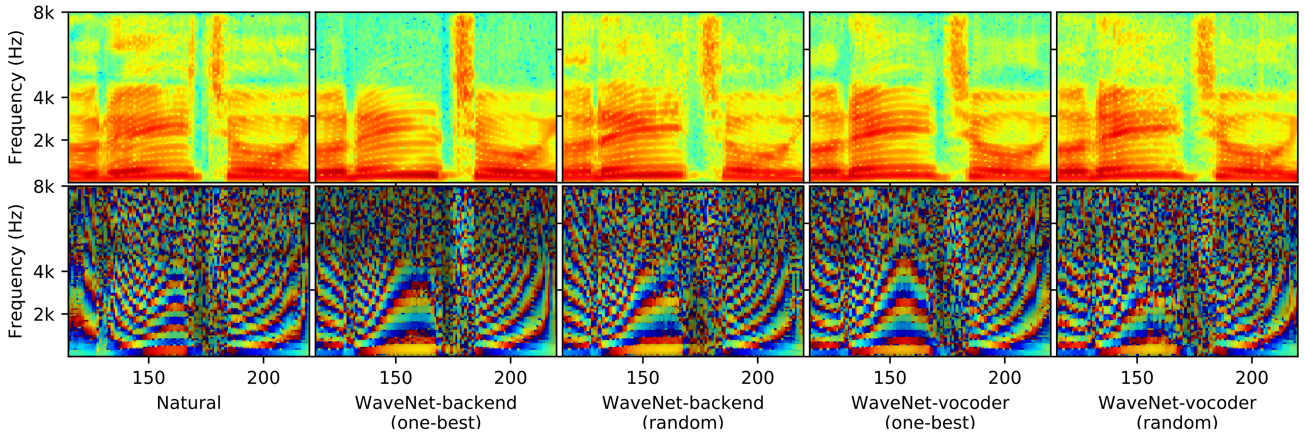

Fig. 6 Spectrogram (row above) and instantaneous frequency (row blow) of natural and generated samples. 'One-best' denotes one-best generation for voiced sound. 'random' denotes random-sampling generation.

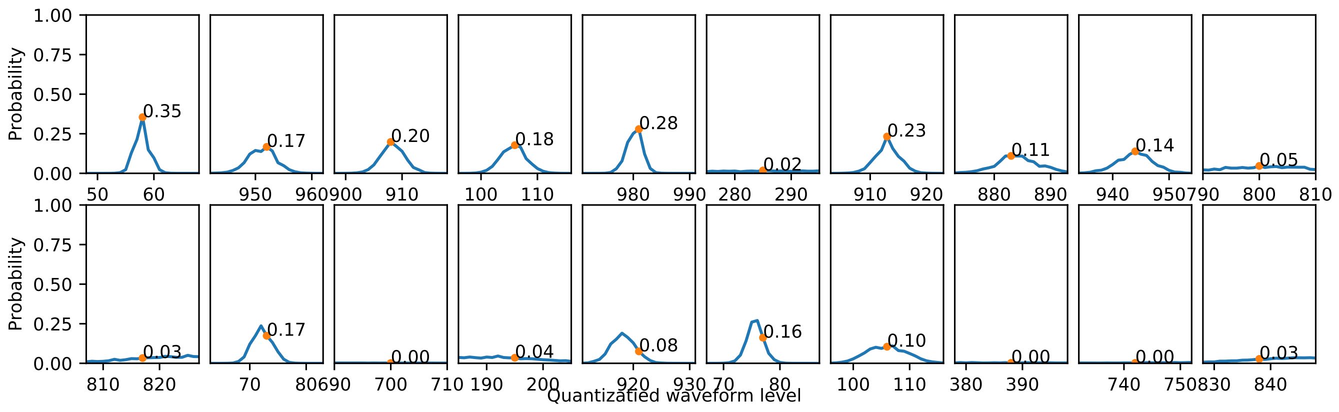

Fig. 7 Distribution $P(\boldsymbol{o}_t|\widehat{\boldsymbol{o}}_{t-R:t-1}, \widehat{\boldsymbol{a}}_{1:N})$ (blue curve) inferred by WaveNet-vocoder and output value $\widehat{\boldsymbol{o}}_t$ (red point) picked from the distribution. Row above shows the case of one-best generation at 10 time steps in a voiced sound. Row below shows the case of random-sampling at the same 10 time steps.

sampling may not pick the value with the highest probability. What's worse, it may randomly select a value that is not highly probable. Compared with random-sampling, the one-best generation method may work better for the voiced sound. As the spectrogram of WaveNet-vocoder in Figure 6 suggests, the harmonic structure of the voiced sound was less noisy. The pattern of instantaneous frequency can be observed and the generated waveform sounds less hoarse.

However, we found that one-best generation method degraded the quality of generated waveform in the case of WaveNet-backend. To explain the possible reasons, we point it out that generating a waveform of length T from WaveNet is a search task in a space of L^T. It is impractical to find the best solution among L^T possible outputs. One-best generation is a greedy search strategy. It can find a good result if the time dependency in the search space is local. As a WaveNet-vocoder may only need to learn the mapping between natural acoustic features and waveforms in local time span, using a greedy search strategy may find the a waveform with an overall high probability (or high quality). However, mapping between linguistic features and waveforms is quite ambiguous in case of a WaveNet-backend. To generate a good waveform, the network may take into consideration previously generated samples over a long time-span. Hence, time dependency is strong and a greedy search method may result in a local optimal solution. Random-sampling may be a better strategy to explore the search space in the case of WaveNet-backend.

4. Conclusion

This report explains the details of WaveNet implemented by the authors, including specific designs on the embedding module and the residual structure inside WaveNet. Based on the implementation, this report shows the observations inside WaveNet. It was found that WaveNet can learn a meaningful manifold for quantized waveforms. Additionally, it was found that the data variance gradually decreased along the WaveNet blocks. Finally, this work introduced a one-best generation method that may be used for WaveNet-vocoder. It was hoped that findings in this work could facilitate further investigation on WaveNet.

References

[1] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W. and Kavukcuoglu, K.: WaveNet: A Generative Model for Raw Audio, *CoRR*, Vol. abs/1609.03499 (online), available from ⟨http://arxiv.org/abs/1609.03499⟩ (2016).

[2] Tamamori, A., Hayashi, T., Kobayashi, K., Takeda, K. and Toda, T.: Speaker-Dependent WaveNet Vocoder, *Proc. Interspeech*, pp. 1118–1122 (online), DOI: 10.21437/Interspeech.2017-314 (2017).

[3] Abadi, M., Agarwal, A. and et. al.: TensorFlow: Large-Scale Machine

Learning on Heterogeneous Distributed Systems (2015).

[4] Tokuda, K., Kobayashi, T., Masuko, T. and Imai, S.: Mel-Generalized Cepstral Analysis A unified approach, *Proc. ICSLP*, pp. 1043–1046 (1994).

[5] Arik, S. O., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Raiman, J., Sengupta, S. et al.: Deep Voice: Real-time Neural Text-to-Speech, *arXiv preprint arXiv:1702.07825* (2017).

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

[7] The HTS Working Group: The Japanese TTS System 'Open JTalk' (2015).

[8] Morise, M., Yokomori, F. and Ozawa, K.: 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).

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

[10] Wang, X., Takaki, S. and Yamagishi, J.: An RNN-based Quantized F0 Model with Multi-tier Feedback Links for Text-to-Speech Synthesis, *Proc. Interspeech*, pp. 1059–1063 (2017).

[11] Maaten, L. v. d. and Hinton, G.: Visualizing data using t-SNE, *Journal of Machine Learning Research*, Vol. 9, No. Nov, pp. 2579–2605 (2008).

[12] He, K., Zhang, X., Ren, S. and Sun, J.: Deep residual learning for image recognition, *Proc. CVPR*, pp. 770–778 (2016).

[13] Greff, K., Srivastava, R. K. and Schmidhuber, J.: Highway and residual networks learn unrolled iterative estimation, *Proc. ICLR* (2017).

[14] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R. et al.: Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions, *arXiv preprint arXiv:1712.05884* (2017).