NEURAL SOURCE-FILTER-BASED WAVEFORM MODEL FOR STATISTICAL PARAMETRIC SPEECH SYNTHESIS

Xin Wang[1], *Shinji Takaki*[1], *Junichi Yamagishi*[1*]

[1]National Institute of Informatics, Japan

wangxin@nii.ac.jp, takaki@nii.ac.jp, jyamagis@nii.ac.jp

ABSTRACT

Neural waveform models such as the WaveNet are used in many recent text-to-speech systems, but the original WaveNet is quite slow in waveform generation because of its autoregressive (AR) structure. Although faster non-AR models were recently reported, they may be prohibitively complicated due to the use of a distilling training method and the blend of other disparate training criteria. This study proposes a non-AR neural source-filter waveform model that can be directly trained using spectrum-based training criteria and the stochastic gradient descent method. Given the input acoustic features, the proposed model first uses a source module to generate a sine-based excitation signal and then uses a filter module to transform the excitation signal into the output speech waveform. Our experiments demonstrated that the proposed model generated waveforms at least 100 times faster than the AR WaveNet and the quality of its synthetic speech is close to that of speech generated by the AR WaveNet. Ablation test results showed that both the sine-wave excitation signal and the spectrum-based training criteria were essential to the performance of the proposed model.

Index Terms— speech synthesis, neural network, waveform modeling

1. INTRODUCTION

Text-to-speech (TTS) synthesis, a technology that converts texts into speech waveforms, has been advanced by using end-to-end architectures [1] and neural-network-based waveform models [2, 3, 4]. Among those waveform models, the WaveNet [2] directly models the distributions of waveform sampling points and has demonstrated outstanding performance. The vocoder version of WaveNet [5], which converts the acoustic features into the waveform, also outperformed other vocoders for the pipeline TTS systems [6].

As an autoregressive (AR) model, the WaveNet is quite slow in waveform generation because it has to generate the waveform sampling points one by one. To improve the generation speed, the Parallel WaveNet [2] and the ClariNet [4] introduce a distilling method to transfer ‘knowledge’ from a teacher AR WaveNet to a student non-AR model that simultaneously generates all the waveform sampling points. However, the concatenation of two large models and the mix of distilling and other training criteria reduce the model interpretability and raise the implementation cost.

In this paper, we propose a neural source-filter waveform model that converts acoustic features into speech waveform. Inspired by classical speech modeling methods [7, 8, 9], we used a source module to generate a sine-wave excitation signal with a specified fundamental frequency (F0). We then used a dilated-convolution-based filter module to transform the sine signal into the speech waveform. The proposed model was trained by minimizing spectral amplitude and phase distances, which can be efficiently implemented using discrete Fourier transforms (DFTs). Because the proposed model is a non-AR model, it generates waveforms much faster than the AR WaveNet. A large-scale listening test showed that the proposed model was close to the AR WaveNet in terms of the Mean opinion score (MOS) on the quality of synthetic speech. An ablation test showed that both the sine-wave excitation and the spectral amplitude distance were crucial to the proposed model.

The model structure and training criteria are explained in Section 2, after which the experiments are described in Section 3.2. Finally, this paper is summarized and concluded in Section 4.

2. PROPOSED MODEL AND TRAINING CRITERIA

2.1. Model structure

The proposed model (shown in Figure 1) converts an input acoustic feature sequence $\boldsymbol{c}_{1:B}$ of length B into a speech waveform $\widehat{\boldsymbol{o}}_{1:T}$ of length T. It includes a source module that generates an excitation signal $\boldsymbol{e}_{1:T}$, a filter module that transforms $\boldsymbol{e}_{1:T}$ into the speech waveform, and a condition module that processes the acoustic features for the source and filter modules. None of these modules takes the previously generated waveform sampling point as the input. Note that the waveform is assumed to be real-valued, i.e., $\widehat{o}_t \in \mathbb{R}, \forall t \in \{1, \cdots, T\}$.

2.1.1. Condition module

The condition module takes as input the acoustic feature sequence $\boldsymbol{c}_{1:B} = \{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_B\}$, where each $\boldsymbol{c}_b = [f_b, \boldsymbol{s}_b^\top]^\top$ contains the F0 f_b and the spectral features $\boldsymbol{s}_b$ of the b-th speech frame. The condition module upsamples the F0 by duplicating f_b to every time step within the b-th frame and then feeds the upsampled F0 sequence $\boldsymbol{f}_{1:T}$ to the source module. Meanwhile, the condition module processes $\boldsymbol{c}_{1:B}$ using a bi-directional recurrent layer with long-short-term memory (LSTM) units [10] and a convolutional layer (CONV), after which the processed features are upsampled and sent to the filter module. This condition module is identical to that of the WaveNet-vocoder trained in our previous study [6].

2.1.2. Source module

Given the input F0 sequence $\boldsymbol{f}_{1:T}$, the source module generates a sine-wave excitation signal $\boldsymbol{e}_{1:T} = \{e_1, \cdots, e_T\}$, where $e_t \in \mathbb{R}, \forall t \in \{1, \cdots, T\}$. Suppose the F0 value of the t-th time step is

*This work was partially supported by JST CREST Grant Number JPMJCR18A6, Japan and by MEXT KAKENHI Grant Numbers (16H06302, 16K16096, 17H04687, 18H04120, 18H04112, 18KT0051), Japan.

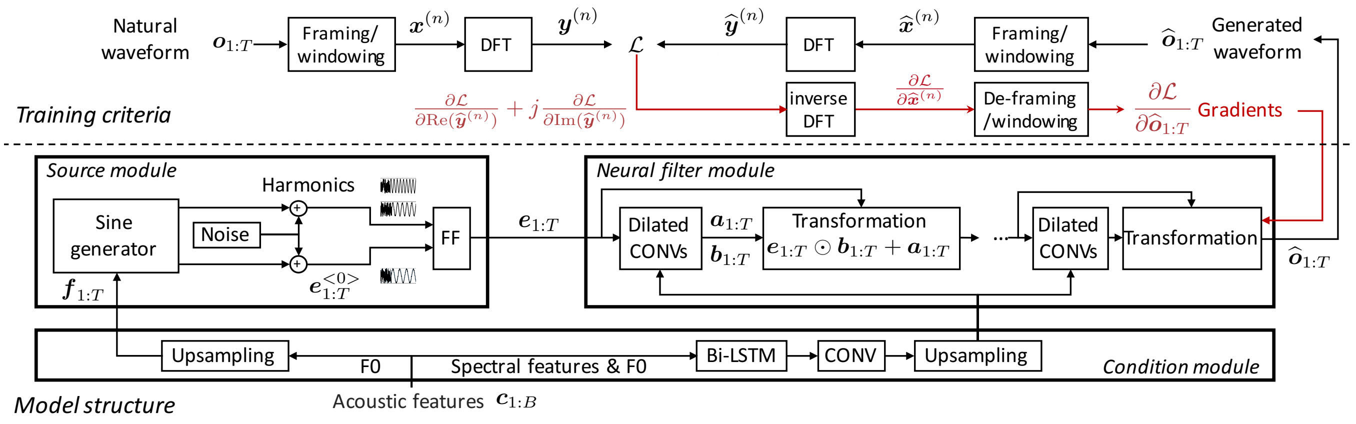


Fig. 1. Structure of proposed model. B and T denote lengths of input feature sequence and output waveform, respectively. FF, CONV, and Bi-LSTM denote feedforward, convolutional, and bi-directional recurrent layers, respectively. DFT denotes discrete Fourier transform.

$f_t \in \mathbb{R}_{\geq 0}$, and $f_t = 0$ denotes being unvoiced. By treating f_t as the instantaneous frequency [11], a signal $e^{<0>}_{1:T}$ can be generated using

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

where $n_t \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian noise, $\phi \in [-\pi, \pi]$ is a random initial phase, and N_s is equal to the waveform sampling rate.

Although we can directly set $e_{1:T} = e^{<0>}_{1:T}$, we tried two additional methods for improvement. First, a 'best' phase ϕ^* for $e^{<0>}_{1:T}$ can be determined in the training stage by maximizing the correlation between $e^{<0>}_{1:T}$ and the natural waveform $o_{1:T}$. During generation, ϕ is still randomly generated. The second method is to generate harmonics by increasing f_k in Equation (1) and use a feedforward (FF) layer to merge the harmonics and $e^{<0>}_{1:T}$ into $e_{1:T}$. In this paper we use 7 harmonics and set $\sigma = 0.003$ and $\alpha = 0.1$.

2.1.3. Neural filter module

Given the excitation signal $e_{1:T}$ from the source module and the processed acoustic features from the condition module, the filter module modulates $e_{1:T}$ using multiple stages of dilated convolution and simple transformations. For example, the first stage takes $e_{1:T}$ and the processed acoustic features as input and produces two signals $a_{1:T}$ and $b_{1:T}$ using dilated convolution. The $e_{1:T}$ is then transformed using $e_{1:T} \odot b_{1:T} + a_{1:T}$, where $\odot$ denotes element-wise multiplication. The transformed signal is further processed in the following stages, and the output of the final stage is used as generated waveform $\widehat{o}_{1:T}$.

The dilated-convolution blocks are similar to those in Parallel WaveNet [2] and ClariNet [4]. Specifically, each block contains multiple dilated-convolution layers with a filter size of 3. The outputs of the convolution layers are merged with the acoustic features provided by the condition module through gated activation functions. After that, the merged features are propagated through the skip channel and transformed into $a_{1:T}$ and $\tilde{b}_{1:T}$. To make sure that $b_{1:T}$ is positive, $b_{1:T}$ is parameterized as $b_{1:T} = \exp(\tilde{b}_{1:T})$.

Because the proposed model does not use the distilling method, it is unnecessary to compute the mean and standard deviation of the transformed signal. Neither is it necessary to form the convolution and transformation blocks as an inverse autoregressive flow [12].

2.2. Training criteria in frequency domain

Because speech perception heavily relies on acoustic cues in the frequency domain, we define training criteria that minimize the spectral amplitude and phase distances, which can be implemented using DFTs. Given these criteria, the proposed model is trained using the stochastic gradient descent (SGD) method. The matrix form of the criteria and the gradients can be found in [13].

2.2.1. Spectral amplitude distance

Following the convention of short-time Fourier analysis, we conduct waveform framing and windowing before producing the spectrum of each frame. For the generated waveform $\widehat{o}_{1:T}$, we use $\widehat{x}^{(n)} = [\widehat{x}^{(n)}_1, \cdots, \widehat{x}^{(n)}_M]^\top \in \mathbb{R}^M$ to denote the n-th waveform frame of length M. We then use $\widehat{y}^{(n)} = [\widehat{y}^{(n)}_1, \cdots, \widehat{y}^{(n)}_K]^\top \in \mathbb{C}^K$ to denote the spectrum of $\widehat{x}^{(n)}$ calculated using K-point DFT. We similarly define $x^{(n)}$ and $y^{(n)}$ for the natural waveform $o_{1:T}$.

Suppose the waveform is sliced into N frames. Then the log spectral amplitude distance is defined as follows:

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

where $\mathtt{Re}(\cdot)$ and $\mathtt{Im}(\cdot)$ denote the real and imaginary parts of a complex number, respectively.

Although $\mathcal{L}_s$ is defined on complex-valued spectra, the gradient $\frac{\partial \mathcal{L}_s}{\partial \widehat{o}_{1:T}} \in \mathbb{R}^T$ for SGD training can be efficiently calculated. Let us consider the n-th frame and compose a complex-valued vector $g^{(n)} = \frac{\partial \mathcal{L}_s}{\partial \mathtt{Re}(\widehat{y}^{(n)})} + j\frac{\partial \mathcal{L}_s}{\partial \mathtt{Im}(\widehat{y}^{(n)})} \in \mathbb{C}^K$, where the k-th element is $g^{(n)}_k = \frac{\partial \mathcal{L}_s}{\partial \mathtt{Re}(\widehat{y}^{(n)}_k)} + j\frac{\partial \mathcal{L}_s}{\partial \mathtt{Im}(\widehat{y}^{(n)}_k)} \in \mathbb{C}$. It can be shown that, as long as $g^{(n)}$ is Hermitian symmetric, the inverse DFT of $g^{(n)}$ is equal to $\frac{\partial \mathcal{L}_s}{\partial \widehat{x}^{(n)}} = [\frac{\partial \mathcal{L}_s}{\partial \widehat{x}^{(n)}_1}, \cdots, \frac{\partial \mathcal{L}_s}{\partial \widehat{x}^{(n)}_m}, \frac{\partial \mathcal{L}_s}{\partial \widehat{x}^{(n)}_M}] \in \mathbb{R}^M$ [1]. Using the same method, $\frac{\partial \mathcal{L}_s}{\partial \widehat{x}^{(n)}}$ for $n \in \{1, \cdots, N\}$ can be computed in parallel. Given $\{\frac{\partial \mathcal{L}_s}{\partial \widehat{x}^{(1)}}, \cdots, \frac{\partial \mathcal{L}_s}{\partial \widehat{x}^{(N)}}\}$, the value of each $\frac{\partial \mathcal{L}_s}{\partial \widehat{o}_t}$ in $\frac{\partial \mathcal{L}_s}{\partial \widehat{o}_{1:T}}$ can be easily accumulated since the relationship between $\widehat{o}_t$ and each $\widehat{x}^{(n)}_m$ has been determined by the framing and windowing operations.

[1]In the implementation using fast Fourier transform, $\widehat{x}^{(n)}$ of length M is zero-padded to length K before DFT. Accordingly, the inverse DFT of $g^{(n)}$ includes the gradients w.r.t. the zero-padded part, which should be discarded.

Table 1. Three framing and DFT configurations for $\mathcal{L}_s$ and $\mathcal{L}_p$

	$\mathcal{L}_{s1}\&\mathcal{L}_{p1}$	$\mathcal{L}_{s2}\&\mathcal{L}_{p2}$	$\mathcal{L}_{s3}\&\mathcal{L}_{p3}$
DFT bins K	512	128	2048
Frame length M	320 (20 ms)	80 (5 ms)	1920 (120 ms)
Frame shift	80 (5 ms)	40 (2.5 ms)	640 (40 ms)

Note: all configurations use Hann window.

In fact, $\frac{\partial \mathcal{L}_s}{\partial \widehat{\boldsymbol{o}}_{1:T}} \in \mathbb{R}^T$ can be calculated in the same manner no matter how we set the framing and DFT configuration, i.e., the values of N, M, and K. Furthermore, multiple $\mathcal{L}_s$s with different configurations can be computed, and the gradients $\frac{\partial \mathcal{L}_s}{\partial \widehat{\boldsymbol{o}}_{1:T}}$ can be simply summed up. For example, using the three $\mathcal{L}_s$s in Table 1 was found to be essential to the proposed model (see Section 3.3).

The Hermitian symmetry of $\boldsymbol{g}^{(n)}$ is satisfied if $\mathcal{L}_s$ is carefully defined. For example, $\mathcal{L}_s$ can be the square error or Kullback-Leibler divergence (KLD) of the spectral amplitudes [14, 15]. The phase distance defined below also satisfies the requirement.

2.2.2. *Phase distance*

Given the spectra, a phase distance is computed as

$$\begin{aligned}\mathcal{L}_p &= \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\left|1-\exp(j(\widehat{\theta}_k^{(n)}-\theta_k^{(n)}))\right|^2\\ &=\sum_{n=1}^{N}\sum_{k=1}^{K}\left[1-\frac{\texttt{Re}(\widehat{y}_k^{(n)})\texttt{Re}(y_k^{(n)})+\texttt{Im}(\widehat{y}_k^{(n)})\texttt{Im}(y_k^{(n)})}{|\widehat{y}_k^{(n)}||y_k^{(n)}|}\right]\end{aligned}, \quad (3)$$

where $\widehat{\theta}_k^{(n)}$ and $\theta_k^{(n)}$ are the phases of $\widehat{y}_k^{(n)}$ and $y_k^{(n)}$, respectively. The gradient $\frac{\partial \mathcal{L}_p}{\partial \widehat{\boldsymbol{o}}_{1:T}}$ can be computed by the same procedure as $\frac{\partial \mathcal{L}_s}{\partial \widehat{\boldsymbol{o}}_{1:T}}$. Multiple $\mathcal{L}_p$s and $\mathcal{L}_s$s with different framing and DFT configurations can be added up as the ultimate training criterion $\mathcal{L}$. For different $\mathcal{L}_*$s, additional DFT/iDFT and framing/windowing blocks should be added to the model in Figure 1.

3. EXPERIMENTS

3.1. Corpus and features

This study used the same Japanese speech corpus and data division recipe as our previous study [16]. This corpus [17] contains neutral reading speech uttered by a female speaker. Both validation and test sets contain 480 randomly selected utterances. Among the 48-hour training data, 9,000 randomly selected utterances (15 hours) were used as the training set in this study. For the ablation test in Section 3.3, the training set was further reduced to 3000 utterances (5 hours). Acoustic features, including 60 dimensions of Mel-generalized cepstral coefficients (MGCs) [18] and 1 dimension of F0, were extracted from the 48 kHz waveforms at a frame shift of 5 ms using WORLD [19]. The natural waveforms were then downsampled to 16 kHz for model training and the listening test.

3.2. Comparison of proposed model, WaveNet, and WORLD

The first experiment compared the four models listed in Table 2[2]. The `WAD` model, which was trained in our previous study [6], contained a condition module, a post-processing module, and 40 dilated CONV blocks, where the k-th CONV block had a dilation size of $2^{\text{modulo}(k,10)}$. `WAC` was similar to `WAD` but used a Gaussian distribution to model the raw waveform at the output layer [4].

[2]The models were implemented using a modified CURRENNT toolkit [20] on a single P100 Nvidia GPU card. Codes, recipes, and generated speech can be found on `https://nii-yamagishilab.github.io`.

Table 2. Models for comparison test in Section 3.2

`WOR`	WORLD vocoder
`WAD`	WaveNet-vocoder for 10-bit discrete μ-law waveform
`WAC`	WaveNet-vocoder using Gaussian dist. for raw waveform
`NSF`	Proposed model for raw waveform

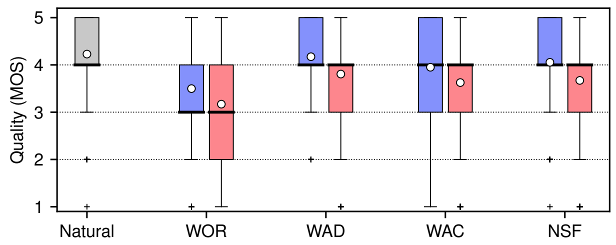


Fig. 2. MOS scores of natural speech, synthetic speech given natural acoustic features (blue boxes), and synthetic speech given generated acoustic features (red boxes). White dots are mean MOS scores.

Table 3. Average number of waveform points generated in 1 s

`WAD`	`NSF` (memory-save mode)	`NSF` (normal mode)
0.19k	20k	227k

The proposed `NSF` contained 5 stages of dilated CONV and transformation, each stage including 10 convolutional layers with a dilation size of $2^{\text{modulo}(k,10)}$ and a filter size of 3. Its condition module was the same as that of `WAD` and `WAC`. `NSF` was trained using $\mathcal{L} = \mathcal{L}_{s1} + \mathcal{L}_{s2} + \mathcal{L}_{s3}$, and the configuration of each $\mathcal{L}_{s*}$ is listed in Table 1. The phase distance $\mathcal{L}_{p*}$ was not used in this test.

Each model generated waveforms using natural and generated acoustic features, where the generated acoustic features were produced by the acoustic models in our previous study [6]. The generated and natural waveforms were then evaluated by paid native Japanese speakers. In each evaluation round the evaluator listened to one speech waveform in each screen and rated the speech quality on a 1-to-5 MOS scale. The evaluator was allowed to replay the waveform and do at most 10 evaluation rounds. The waveforms in an evaluation round were for the same text and were played in a random order. Note that the waveforms generated from `NSF` and `WAC` were converted to 16-bit PCM format before evaluation.

A total of 245 evaluators conducted 1444 valid evaluation rounds in all, and the results are plotted in Figure 2. Two-sided Mann-Whitney tests showed that the difference between any pair of models is statistically significant ($p < 0.01$) except `NSF` and `WAC` when the two models used generated acoustic features. In general, `NSF` outperformed `WOR` and `WAC` but performed slightly worse than `WAD`. The gap of the mean MOS scores between `NSF` and `WAD` was about 0.12, given either natural or generated acoustic features. A possible reason for this result may be the difference between the non-AR and AR model structures, which is similar to the difference between the finite and infinite impulse response filters. `WAC` performed worse than `WAD` because some syllables were perceived to be trembling in pitch, which may be caused by the random sampling generation method. `WAD` alleviated this artifact by using a one-best generation method in voiced regions [6].

After the MOS test, we compared the waveform generation speed of `NSF` and `WAD`. The implementation of `NSF` has a normal

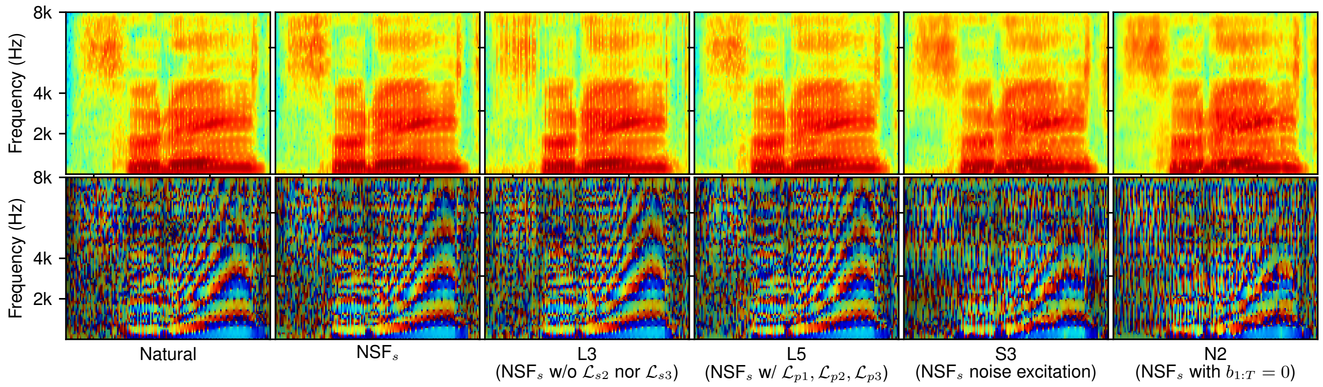


Fig. 3. Spectrogram (top) and instantaneous frequency (bottom) of natural waveform and waveforms generated from models in Table 4 given natural acoustic features in test set (utterance AOZORAR_03372_T01). Figures are plotted using 5 ms frame length and 2.5 ms frame shift.

Table 4. Models for ablation test (Section 3.3)

NSF$_s$	NSF trained on 5-hour data
L1	NSF$_s$ without using $\mathcal{L}_{s3}$ (i.e., $\mathcal{L} = \mathcal{L}_{s1} + \mathcal{L}_{s2}$)
L2	NSF$_s$ without using $\mathcal{L}_{s2}$ (i.e., $\mathcal{L} = \mathcal{L}_{s1} + \mathcal{L}_{s3}$)
L3	NSF$_s$ without using $\mathcal{L}_{s2}$ nor $\mathcal{L}_{s3}$ (i.e., $\mathcal{L} = \mathcal{L}_{s1}$)
L4	NSF$_s$ using $\mathcal{L} = \mathcal{L}_{s1} + \mathcal{L}_{s2} + \mathcal{L}_{s3} + \mathcal{L}_{p1} + \mathcal{L}_{p2} + \mathcal{L}_{p3}$
L5	NSF$_s$ using KLD of spectral amplitudes
S1	NSF$_s$ without harmonics
S2	NSF$_s$ without harmonics or 'best' phase ϕ^*
S3	NSF$_s$ only using noise as excitation
N1	NSF$_s$ with $\boldsymbol{b}_{1:T} = 1$ in filter's transformation layers
N2	NSF$_s$ with $\boldsymbol{b}_{1:T} = 0$ in filter's transformation layers

generation mode and a memory-save one. The normal mode allocates all the required GPU memory once but cannot generate waveforms longer than 6 seconds because of the insufficient memory space in a single GPU card. The memory-save mode can generate long waveforms because it releases and allocates the memory layer by layer, but the repeated memory operations are time consuming.

We evaluated NSF using both modes on a smaller test set, in which each of the 80 generated test utterances was around 5 seconds long. As the results in Table 3 show, NSF is much faster than WAD. Note that WAD allocates and re-uses a small size of GPU memory, which needs no repeated memory operation. WAD is slow mainly because of the AR generation process. Of course, both WAD and NSF can be improved if our toolkit is further optimized. Particularly, if the memory operation can be speeded up, the memory-save mode of NSF will be much faster.

3.3. Ablation test on proposed model

This experiment was an ablation test on NSF. Specifically, the 11 variants of NSF listed in Table 4 were trained using the 5-hour training set. For a fair comparison, NSF was re-trained using the 5-hour data, and this variant is referred to as NSF$_s$. The speech waveforms were generated given the natural acoustic features and rated in 1444 evaluation rounds by the same group of evaluators in Section 3.2. This test excluded natural waveform for evaluation.

The results are plotted in Figure 4. The difference between NST$_s$ and any other model except S2 was statistically significant ($p < 0.01$). Comparison among NST$_s$, L1, L2, and L3 shows that using multiple $\mathcal{L}_s$s listed in Table 1 is beneficial. For L3, which used only $\mathcal{L}_{s1}$, the generated waveform points tended to concentrate in each short period, and the waveform sounded like a pulse train especially in unvoiced part. This can be observed from Figure 3

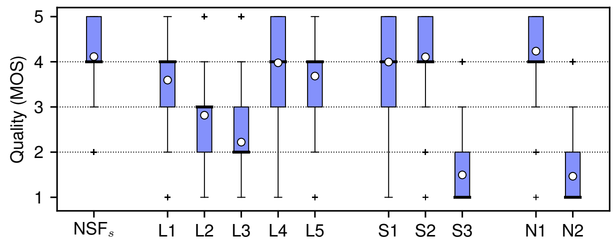


Fig. 4. MOS scores of synthetic samples from NSF$_s$ and its variants given natural acoustic features. White dots are mean MOS scores.

where the spectrogram was plotted using a small frame length (5 ms). Accordingly, this artifact can be alleviated by adding $\mathcal{L}_{s2}$ with a frame length of 5 ms for model training, which explained the improvement in L1. Using phase distance (L4) didn't improve the speech quality even though the value of the phase distance was consistently decreased on both training and validation data.

The good result of S2 indicates that a single sine-wave excitation with a random initial phase also works. Without the sine-wave excitation, S3 generated waveforms that were intelligible but lacked stable harmonic structure. N1 slightly outperformed NSF$_s$ while N2 produced unstable harmonic structures. Because the transformation in N1 is equivalent to skip-connection [21], the result indicates that the skip-connection may help the model training.

4. CONCLUSION

In this paper, we proposed a neural waveform model with separated source and filter modules. The source module produces a sine-wave excitation signal with a specified F0, and the filter module uses dilated convolution to transform the excitation into a waveform. Our experiment demonstrated that the sine-wave excitation was essential for generating waveforms with harmonic structures. We also found that multiple spectral-based training criteria and the transformation in the filter module contributed to the performance of the proposed model. Compared with the AR WaveNet, the proposed model generated speech with a similar quality at a much faster speed.

The proposed model is an initial trial and thus can be improved in many aspects. For example, it may be better to separate the filters for the harmonic and noise parts [9]. Source signals with glottal features [22, 23] can also be tried. It is also possible to model waveforms at a higher sampling rate. When applying the model to convert linguistic features into the waveform, we observed the over-smoothing affect in the high-frequency speech band, which indicates that the distilling or generative adversarial approach [24] is needed.

5. REFERENCES

[1] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al., "Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions," in *Proc. ICASSP*, 2018, pp. 4779–4783.

[2] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis, "Parallel WaveNet: Fast high-fidelity speech synthesis," in *Proc. ICML*, 2018, pp. 3918–3926.

[3] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al., "Parallel WaveNet: Fast high-fidelity speech synthesis," *arXiv preprint arXiv:1711.10433*, 2017.

[4] Wei Ping, Kainan Peng, and Jitong Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," *arXiv preprint arXiv:1807.07281*, 2018.

[5] Akira Tamamori, Tomoki Hayashi, Kazuhiro Kobayashi, Kazuya Takeda, and Tomoki Toda, "Speaker-dependent WaveNet vocoder," in *Proc. Interspeech*, 2017, pp. 1118–1122.

[6] Xin Wang, Jaime Lorenzo-Trueba, Shinji Takaki, Lauri Juvela, and Junichi Yamagishi, "A comparison of recent waveform generation and acoustic modeling methods for neural-network-based speech synthesis," in *Proc. ICASSP*, 2018, pp. 4804–4808.

[7] Per Hedelin, "A tone oriented voice excited vocoder," in *Proc. ICASSP*. IEEE, 1981, vol. 6, pp. 205–208.

[8] Robert McAulay and Thomas Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.

[9] Yannis Stylianou, *Harmonic plus noise models for speech, combined with statistical methods, for speech and speaker modification*, Ph.D. thesis, Ecole Nationale Superieure des Telecommunications, 1996.

[10] Alex Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Ph.D. thesis, Technische Universität München, 2008.

[11] John R Carson and Thornton 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.

[12] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling, "Improved variational inference with inverse autoregressive flow," in *Proc. NIPS*, 2016, pp. 4743–4751.

[13] Shinji Takaki, Toru Nakashika, Xin Wang, and Junichi Yamagishi, "STFT spectral loss for training a neural speech waveform model," in *Proc. ICASSP (submitted)*, 2018.

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

[15] Shinji Takaki, Hirokazu Kameoka, and Junichi 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.

[16] Hieu-Thi Luong, Xin Wang, Junichi Yamagishi, and Nobuyuki Nishizawa, "Investigating accuracy of pitch-accent annotations in neural-network-based speech synthesis and denoising effects," in *Proc. Interspeech*, 2018, pp. 37–41.

[17] Hisashi Kawai, Tomoki Toda, Jinfu Ni, Minoru Tsuzaki, and Keiichi Tokuda, "XIMERA: A new TTS from ATR based on corpus-based technologies," in *Proc. SSW5*, 2004, pp. 179–184.

[18] Keiichi Tokuda, Takao Kobayashi, Takashi Masuko, and Satoshi Imai, "Mel-generalized cepstral analysis a unified approach," in *Proc. ICSLP*, 1994, pp. 1043–1046.

[19] Masanori Morise, Fumiya Yokomori, and Kenji 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.

[20] Felix Weninger, Johannes Bergmann, and Björn 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.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[22] Gunnar Fant, Johan Liljencrants, and Qi-guang Lin, "A four-parameter model of glottal flow," *STL-QPSR*, vol. 4, no. 1985, pp. 1–13, 1985.

[23] Lauri Juvela, Bajibabu Bollepalli, Manu Airaksinen, and Paavo Alku, "High-pitched excitation generation for glottal vocoding in statistical parametric speech synthesis using a deep neural network," in *Proc. ICASSP*, 2016, pp. 5120–5124.

[24] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.

A. FORWARD COMPUTATION

Figure 5 plots the two steps to derive the spectrum from the generated waveform $\widehat{\boldsymbol{o}}_{1:T}$. We use $\widehat{\boldsymbol{x}}^{(n)} = [\widehat{x}_1^{(n)}, \cdots, \widehat{x}_M^{(n)}]^\top \in \mathbb{R}^M$ to denote the n-th waveform frame of length M. 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, i.e., $\widehat{\boldsymbol{y}}^{(n)} = \text{DFT}_K(\widehat{\boldsymbol{x}}^{(n)})$. For fast Fourier transform, K is set to the power of 2, and $\widehat{\boldsymbol{x}}^{(n)}$ is zero-padded to length K before DFT.

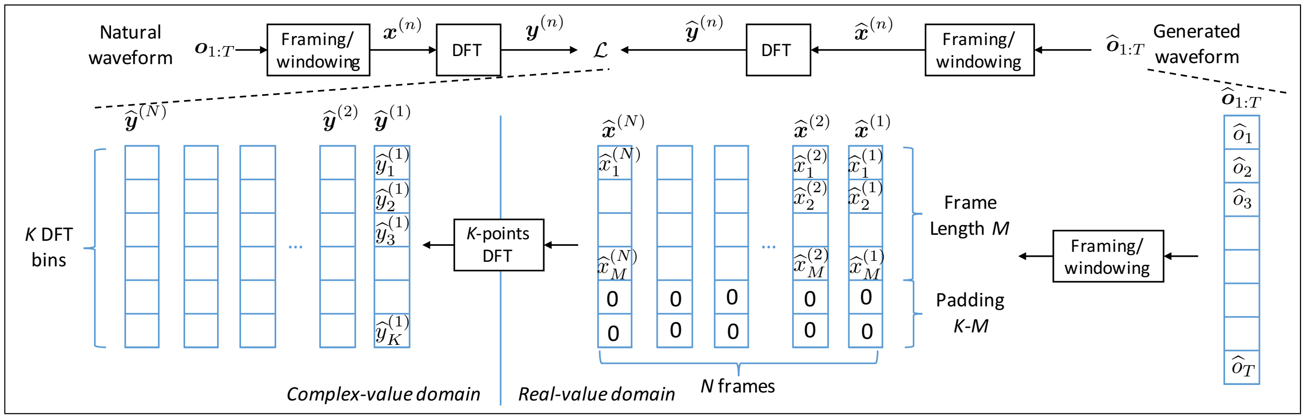


Fig. 5. Framing/windowing and DFT steps. T, M, K denotes waveform length, frame length, and number of DFT bins.

A.1. Framing and windowing

The framing and windowing operation is also parallelized over $\widehat{x}_m^{(n)}$ using `for_each` command in CUDA/Thrust. However, for explanation, let's use the matrix operation in Figure 6. In other words, we compute

$$\widehat{x}_m^{(n)} = \sum_{t=1}^{T} \widehat{o}_t w_t^{(n,m)}, \tag{4}$$

where $w_t^{(n,m)}$ is the element in the $\big[(n-1)\times M + m\big]$-th row and the t-th column of the transformation matrix $\boldsymbol{W}$.

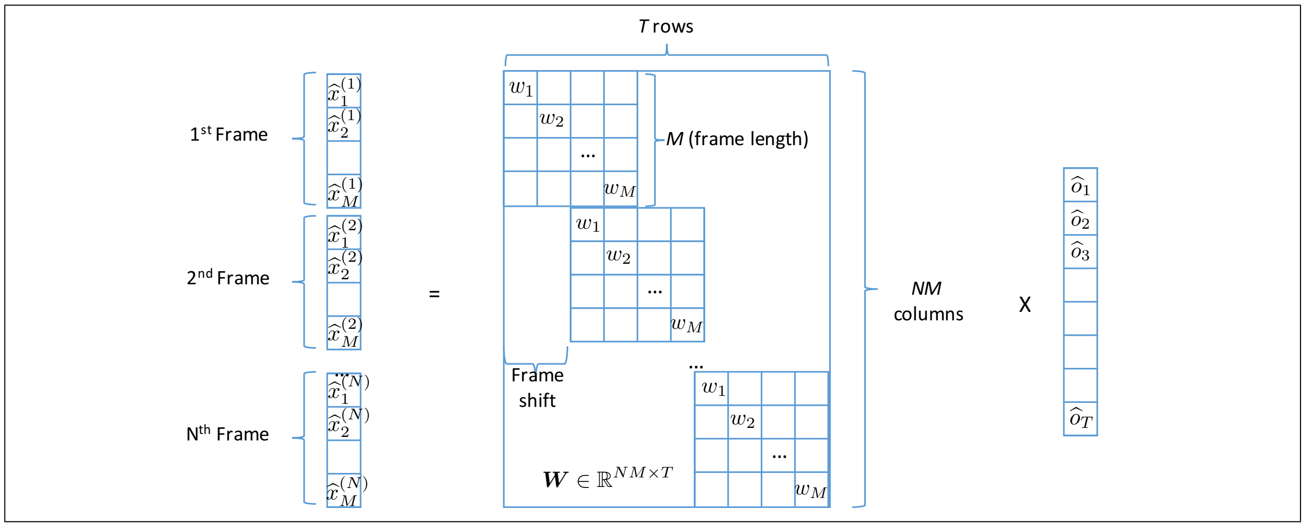


Fig. 6. A matrix format of framing/windowing operation, where $\boldsymbol{w}_{1:M}$ denote coefficients of Hann window.

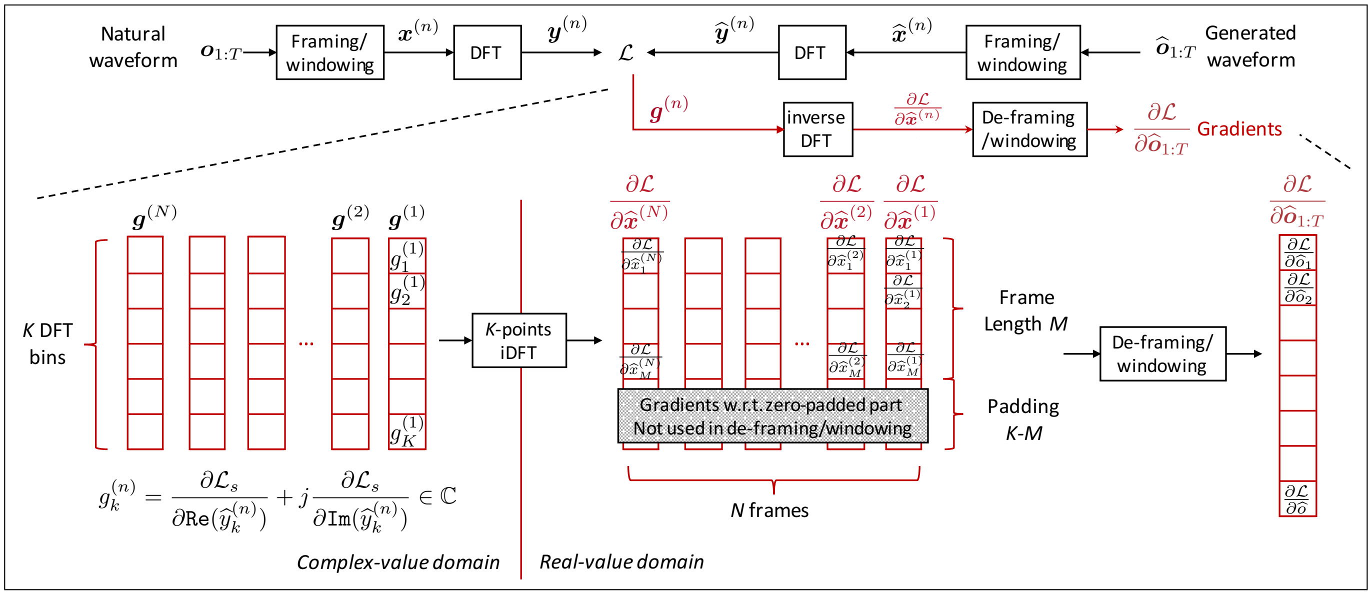
A.2. DFT

Our implementation uses cuFFT (`cufftExecR2C` and `cufftPlan1d`) [3] to compute $\{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(N)}\}$ in parallel.

[3] https://docs.nvidia.com/cuda/cufft/index.html

B. BACKWARD COMPUTATION

For back-propagation, we need to compute the gradient $\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{o}}_{1:T}} \in \mathbb{R}^T$ following the steps plotted in Figure 7.



Natural waveform
$\boldsymbol{o}_{1:T}$
Framing/ windowing
$\boldsymbol{x}^{(n)}$
DFT
$\boldsymbol{y}^{(n)}$
$\mathcal{L}$
$\widehat{\boldsymbol{y}}^{(n)}$
DFT
$\widehat{\boldsymbol{x}}^{(n)}$
Framing/ windowing
$\widehat{\boldsymbol{o}}_{1:T}$
Generated waveform
$\boldsymbol{g}^{(n)}$
inverse DFT
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{x}}^{(n)}}$
De-framing /windowing
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{o}}_{1:T}}$
Gradients
$\boldsymbol{g}^{(N)}$
$\boldsymbol{g}^{(2)}$
$\boldsymbol{g}^{(1)}$
$g_1^{(1)}$
$g_2^{(1)}$
$g_K^{(1)}$
K DFT bins
K-points iDFT
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{x}}^{(N)}}$
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{x}}^{(2)}}$
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{x}}^{(1)}}$
Gradients w.r.t. zero-padded part
Not used in de-framing/windowing
Frame Length M
Padding K-M
De-framing/ windowing
N frames
$\frac{\partial \mathcal{L}}{\partial \widehat{o}_1}$
$\frac{\partial \mathcal{L}}{\partial \widehat{o}_2}$
$g_k^{(n)} = \frac{\partial \mathcal{L}_s}{\partial \mathrm{Re}(\widehat{y}_k^{(n)})} + j\frac{\partial \mathcal{L}_s}{\partial \mathrm{Im}(\widehat{y}_k^{(n)})} \in \mathbb{C}$
Complex-value domain
Real-value domain

Fig. 7. Steps to compute gradients

B.1. The 2nd step: from $\frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}}$ to $\frac{\partial \mathcal{L}}{\partial \widehat{o}_t}$

Suppose we have $\{\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{x}}^{(1)}}, \cdots, \frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{x}}^{(N)}}\}$, where each $\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{x}}^{(n)}} \in \mathbb{R}^M$ and $\frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}} \in \mathbb{R}$. Then, we can compute $\frac{\partial \mathcal{L}}{\partial \widehat{o}_t}$ on the basis Equation (4) as

$$\frac{\partial \mathcal{L}}{\partial \widehat{o}_t} = \sum_{n=1}^{N}\sum_{m=1}^{M} \frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}} w_t^{(n,m)}, \tag{5}$$

where $w_t^{(n,m)}$ are the framing/windowing coefficients. This equation explains what we mean by saying '$\frac{\partial \mathcal{L}}{\partial \widehat{o}_t}$ can be easily accumulated the relationship between $\widehat{o}_t$ and each $\widehat{x}_m^{(n)}$ has been determined by the framing and windowing operations'.

Our implementation uses CUDA/Thrust `for_each` command to launch T threads and compute $\frac{\partial \mathcal{L}}{\partial \widehat{o}_t}, t \in \{1, \cdots, T\}$ in parallel. Because $\widehat{o}_t$ is only used in a few frames and there is only one $w_t^{(n,m)} \neq 0$ for each $\{n, t\}$, Equation (5) can be optimized as

$$\frac{\partial \mathcal{L}}{\partial \widehat{o}_t} = \sum_{n=N_{t,min}}^{N_{t,max}} \frac{\partial \mathcal{L}}{\partial \widehat{x}_{m_{t,n}}^{(n)}} w_t^{(n,m_{t,n})}, \tag{6}$$

where $[N_{t,min}, N_{t,max}]$ is the frame range that $\widehat{o}_t$ appears, and $m_{t,n}$ is the position of $\widehat{o}_t$ in the n-th frame.

B.2. The 1st step: compute $\frac{\partial \mathcal{L}}{\partial \widehat{x}_m^{(n)}}$

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

$$\mathtt{Re}(\widehat{y}_k^{(n)}) = \sum_{m=1}^{M} \widehat{x}_m^{(n)} \cos(\frac{2\pi}{K}(k-1)(m-1)), \tag{7}$$

$$\mathtt{Im}(\widehat{y}_k^{(n)}) = -\sum_{m=1}^{M} \widehat{x}_m^{(n)} \sin(\frac{2\pi}{K}(k-1)(m-1)), \tag{8}$$

where $k \in [1, K]$. Note that, although the sum should be $\sum_{m=1}^{K}$, the summation over the zero-padded part $\sum_{m=M+1}^{K} 0\cos(\frac{2\pi}{K}(k-1)(m-1))$ can be safely ignored [4].

[4]Although we can avoid zero-padding by setting $K = M$, in practice K is usually the power of 2 while the frame length M is not.

Suppose we compute a log spectral amplitude distance $\mathcal{L}$ over the N frames as

$$\mathcal{L} = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\Big[\log\frac{\mathtt{Re}(y_k^{(n)})^2 + \mathtt{Im}(y_k^{(n)})^2}{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2}\Big]^2. \tag{9}$$

Because $\mathcal{L}$, $\widehat{x}_m^{(n)}$, $\mathtt{Re}(\widehat{y}_k^{(n)})$, and $\mathtt{Im}(\widehat{y}_k^{(n)})$ are real-valued numbers, we can compute the gradient $\frac{\partial\mathcal{L}}{\partial\widehat{x}_m^{(n)}}$ using the chain rule:

$$\frac{\partial\mathcal{L}}{\partial\widehat{x}_m^{(n)}} = \sum_{k=1}^{K}\frac{\partial\mathcal{L}}{\partial\mathtt{Re}(\widehat{y}_k^{(n)})}\frac{\partial\mathtt{Re}(\widehat{y}_k^{(n)})}{\partial\widehat{x}_m^{(n)}} + \sum_{k=1}^{K}\frac{\partial\mathcal{L}}{\partial\mathtt{Im}(\widehat{y}_k^{(n)})}\frac{\partial\mathtt{Im}(\widehat{y}_k^{(n)})}{\partial\widehat{x}_m^{(n)}} \tag{10}$$

$$= \sum_{k=1}^{K}\frac{\partial\mathcal{L}}{\partial\mathtt{Re}(\widehat{y}_k^{(n)})}\cos(\frac{2\pi}{K}(k-1)(m-1)) - \sum_{k=1}^{K}\frac{\partial\mathcal{L}}{\partial\mathtt{Im}(\widehat{y}_k^{(n)})}\sin(\frac{2\pi}{K}(k-1)(m-1)). \tag{11}$$

Once we compute $\frac{\partial\mathcal{L}}{\partial\widehat{x}_m^{(n)}}$ for each m and n, we can use Equation (6) to compute the gradient $\frac{\partial\mathcal{L}}{\partial\widehat{o}_t}$.

B.3. Implementation of the 1st step using inverse DFT

Because $\frac{\partial\mathcal{L}}{\partial\mathtt{Re}(\widehat{y}_k^{(n)})}$ and $\frac{\partial\mathcal{L}}{\partial\mathtt{Im}(\widehat{y}_k^{(n)})}$ are real numbers, we can directly implement Equation (11) using matrix multiplication. However, a more efficient way is to use inverse DFT (iDFT).

Suppose a complex valued signal $\boldsymbol{g} = [g_1, g_2, \cdots, g_L] \in \mathbb{C}^K$, we compute $\boldsymbol{b} = [b_1, \cdots, b_K]$ as the K-point inverse DFT of $\boldsymbol{g}$ by [5]

$$b_m = \sum_{k=1}^{K} g_k e^{j\frac{2\pi}{K}(k-1)(m-1)} \tag{12}$$

$$= \sum_{k=1}^{K}[\mathtt{Re}(g_k) + j\mathtt{Im}(g_k)][\cos(\frac{2\pi}{K}(k-1)(m-1)) + j\sin(\frac{2\pi}{K}(k-1)(m-1))] \tag{13}$$

$$= \sum_{k=1}^{K}\mathtt{Re}(g_k)\cos(\frac{2\pi}{K}(k-1)(m-1)) - \sum_{k=1}^{K}\mathtt{Im}(g_k)\sin(\frac{2\pi}{K}(k-1)(m-1)) \tag{14}$$

$$+ j\Big[\sum_{k=1}^{K}\mathtt{Re}(g_k)\sin(\frac{2\pi}{K}(k-1)(m-1)) + \sum_{k=1}^{K}\mathtt{Im}(g_k)\cos(\frac{2\pi}{K}(k-1)(m-1))\Big]. \tag{15}$$

For the first term in Line 15, we can write

$$\sum_{k=1}^{K}\mathtt{Re}(g_k)\sin(\frac{2\pi}{K}(k-1)(m-1)) \tag{16}$$

$$=\mathtt{Re}(g_0)\sin(\frac{2\pi}{K}(1-1)(m-1)) + \mathtt{Re}(g_{\frac{K}{2}+1})\sin(\frac{2\pi}{K}(\frac{K}{2}+1-1)(m-1)) \tag{17}$$

$$+ \sum_{k=2}^{\frac{K}{2}}\mathtt{Re}(g_k)\sin(\frac{2\pi}{K}(k-1)(m-1)) + \sum_{k=\frac{K}{2}+2}^{K}\mathtt{Re}(g_k)\sin(\frac{2\pi}{K}(k-1)(m-1)) \tag{18}$$

$$=\sum_{k=2}^{\frac{K}{2}}\Big[\mathtt{Re}(g_k)\sin(\frac{2\pi}{K}(k-1)(m-1)) + \mathtt{Re}(g_{(K+2-k)})\sin(\frac{2\pi}{K}(K+2-k-1)(m-1))\Big] \tag{19}$$

$$=\sum_{k=2}^{\frac{K}{2}}\Big[\mathtt{Re}(g_k) - \mathtt{Re}(g_{(K+2-k)})\Big]\sin(\frac{2\pi}{K}(k-1)(m-1)) \tag{20}$$

Note that in Line (17), $\mathtt{Re}(g_1)\sin(\frac{2\pi}{K}(1-1)(m-1)) = \mathtt{Re}(g_1)\sin(0) = 0$, and $\mathtt{Re}(g_{\frac{K}{2}+1})\sin(\frac{2\pi}{K}(\frac{K}{2}+1-1)(m-1)) = \mathtt{Re}(g_{\frac{K}{2}+1})\sin((m-1)\pi) = 0$.

It is easy to those that Line (20) is equal to 0 if $\mathtt{Re}(g_k) = \mathtt{Re}(g_{(K+2-k)})$, for any $k \in [2, \frac{K}{2}]$. Similarly, it can be shown that $\sum_{k=1}^{K}\mathtt{Im}(g_k)\cos(\frac{2\pi}{K}(k-1)(m-1)) = 0$ if $\mathtt{Im}(g_k) = -\mathtt{Im}(g_{(K+2-k)}), k \in [2, \frac{K}{2}]$ and $\mathtt{Im}(g_1) = \mathtt{Im}(g_{(\frac{K}{2}+1)}) = 0$. When these two terms are equal to 0, the imaginary part in Line (15) will be 0, and $b_m = \sum_{k=1}^{K} g_k e^{j\frac{2\pi}{K}(k-1)(m-1)}$ in Line (12) will be a real number.

[5] cuFFT performs un-normalized FFTs, i.e., the scaling factor $\frac{1}{K}$ is not used.

To summarize, if $\boldsymbol{g}$ satisfies the conditions below

$$\texttt{Re}(g_k) = \texttt{Re}(g_{(K+2-k)}), \quad k \in [2, \frac{K}{2}] \tag{21}$$

$$\texttt{Im}(g_k) = \begin{cases} -\texttt{Im}(g_{(K+2-k)}), & k \in [2, \frac{K}{2}] \\ 0, & k = \{1, \frac{K}{2}+1\} \end{cases} \tag{22}$$

inverse DFT of $\boldsymbol{g}$ will be real-valued:

$$\sum_{k=1}^{K} g_k e^{j\frac{2\pi}{K}(k-1)(m-1)} = \sum_{k=1}^{K} \texttt{Re}(g_k)\cos(\frac{2\pi}{K}(k-1)(m-1)) - \sum_{k=1}^{K} \texttt{Im}(g_k)\sin(\frac{2\pi}{K}(k-1)(m-1)) \tag{23}$$

This is a basic concept in signal processing: the iDFT of a conjugate-symmetric (Hermitian)[6] signal will be a real-valued signal.

We can observer from Equation (23) and (11) that, if $\Big[\frac{\partial\mathcal{L}}{\partial\texttt{Re}(\widehat{y}_1^{(n)})} + j\frac{\partial\mathcal{L}}{\partial\texttt{Im}(\widehat{y}_1^{(n)})}, \cdots, \frac{\partial\mathcal{L}}{\partial\texttt{Re}(\widehat{y}_K^{(n)})} + j\frac{\partial\mathcal{L}}{\partial\texttt{Im}(\widehat{y}_K^{(n)})}\Big]^\top$ is conjugate-symmetric, the gradient vector $\frac{\partial\mathcal{L}}{\partial\widehat{\boldsymbol{x}}^{(n)}} = [\frac{\partial\mathcal{L}}{\partial\widehat{x}_1^{(n)}}, \cdots, \frac{\partial\mathcal{L}}{\partial\widehat{x}_M^{(n)}}]^\top$ be computed using iDFT:

$$\begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\widehat{x}_1^{(n)}} \\ \frac{\partial\mathcal{L}}{\partial\widehat{x}_2^{(n)}} \\ \cdots \\ \frac{\partial\mathcal{L}}{\partial\widehat{x}_M^{(n)}} \\ \frac{\partial\mathcal{L}}{\partial\widehat{x}_{M+1}^{(n)}} \\ \cdots \\ \frac{\partial\mathcal{L}}{\partial\widehat{x}_K^{(n)}} \end{bmatrix} = \text{iDFT}(\begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\texttt{Re}(\widehat{y}_1^{(n)})} + j\frac{\partial\mathcal{L}}{\partial\texttt{Im}(\widehat{y}_1^{(n)})} \\ \frac{\partial\mathcal{L}}{\partial\texttt{Re}(\widehat{y}_2^{(n)})} + j\frac{\partial\mathcal{L}}{\partial\texttt{Im}(\widehat{y}_2^{(n)})} \\ \cdots \\ \frac{\partial\mathcal{L}}{\partial\texttt{Re}(\widehat{y}_M^{(n)})} + j\frac{\partial\mathcal{L}}{\partial\texttt{Im}(\widehat{y}_M^{(n)})} \\ \frac{\partial\mathcal{L}}{\partial\texttt{Re}(\widehat{y}_{M+1}^{(n)})} + j\frac{\partial\mathcal{L}}{\partial\texttt{Im}(\widehat{y}_{M+1}^{(n)})} \\ \cdots \\ \frac{\partial\mathcal{L}}{\partial\texttt{Re}(\widehat{y}_K^{(n)})} + j\frac{\partial\mathcal{L}}{\partial\texttt{Im}(\widehat{y}_K^{(n)})} \end{bmatrix}). \tag{24}$$

Note that $\{\frac{\partial\mathcal{L}}{\partial\widehat{x}_{M+1}^{(n)}}, \cdots, \frac{\partial\mathcal{L}}{\partial\widehat{x}_K^{(n)}}\}$ are the gradients w.r.t to the zero-padded part, which will not be used and can safely set to 0. The iDFT of a conjugate symmetric signal can be executed using cuFFT `cufftExecC2R` command. It is more efficient than other implementations of Equation (11) because

- there is no need to compute the imaginary part;
- there is no need to compute and allocate GPU memory for g_k where $k \in [\frac{K}{2}+2, K]$ because of the conjugate symmetry;
- iDFT can be executed for all the N frames in parallel.

B.4. Conjugate symmetry complex-valued gradient vector

The conjugate symmetry of $\Big[\frac{\partial\mathcal{L}}{\partial\texttt{Re}(\widehat{y}_1^{(n)})} + j\frac{\partial\mathcal{L}}{\partial\texttt{Im}(\widehat{y}_1^{(n)})}, \cdots, \frac{\partial\mathcal{L}}{\partial\texttt{Re}(\widehat{y}_K^{(n)})} + j\frac{\partial\mathcal{L}}{\partial\texttt{Im}(\widehat{y}_K^{(n)})}\Big]^\top$ is satisfied if $\mathcal{L}$ is carefully chosen. Luckily, most of the common distance metrics can be used.

B.4.1. Log spectral amplitude distance

Given the log spectral amplitude distance $\mathcal{L}_s$ in Equation (9), we can compute

$$\frac{\partial\mathcal{L}_s}{\partial\texttt{Re}(\widehat{y}_k^{(n)})} = \Big[\log[\texttt{Re}(\widehat{y}_k^{(n)})^2 + \texttt{Im}(\widehat{y}_k^{(n)})^2] - \log[\texttt{Re}(y_k^{(n)})^2 + \texttt{Im}(y_k^{(n)})^2]\Big]\frac{2\texttt{Re}(\widehat{y}_k^{(n)})}{\texttt{Re}(\widehat{y}_k^{(n)})^2 + \texttt{Im}(\widehat{y}_k^{(n)})^2} \tag{25}$$

$$\frac{\partial\mathcal{L}_s}{\partial\texttt{Im}(\widehat{y}_k^{(n)})} = \Big[\log[\texttt{Re}(\widehat{y}_k^{(n)})^2 + \texttt{Im}(\widehat{y}_k^{(n)})^2] - \log[\texttt{Re}(y_k^{(n)})^2 + \texttt{Im}(y_k^{(n)})^2]\Big]\frac{2\texttt{Im}(\widehat{y}_k^{(n)})}{\texttt{Re}(\widehat{y}_k^{(n)})^2 + \texttt{Im}(\widehat{y}_k^{(n)})^2} \tag{26}$$

Because $\widehat{\boldsymbol{y}}^{(n)}$ is the DFT spectrum of the real-valued signal, $\widehat{\boldsymbol{y}}^{(n)}$ is conjugate symmetric, and $\texttt{Re}(\widehat{y}_k^{(n)})$ and $\texttt{Im}(\widehat{y}_k^{(n)})$ satisfy the condition in Equations (21) and (22), respectively. Because the amplitude $\texttt{Re}(\widehat{y}_k^{(n)})^2 + \texttt{Im}(\widehat{y}_k^{(n)})^2$ does not change the symmetry, $\frac{\partial\mathcal{L}_s}{\partial\texttt{Re}(\widehat{y}_k^{(n)})}$ and $\frac{\partial\mathcal{L}_s}{\partial\texttt{Im}(\widehat{y}_k^{(n)})}$ also satisfy the conditions in Equations (21) and (22), respectively, and $\Big[\frac{\partial\mathcal{L}_s}{\partial\texttt{Re}(\widehat{y}_1^{(n)})} + j\frac{\partial\mathcal{L}_s}{\partial\texttt{Im}(\widehat{y}_1^{(n)})}, \cdots, \frac{\partial\mathcal{L}_s}{\partial\texttt{Re}(\widehat{y}_K^{(n)})} + j\frac{\partial\mathcal{L}_s}{\partial\texttt{Im}(\widehat{y}_K^{(n)})}\Big]^\top$ is conjugate-symmetric.

[6]It should be called circular conjugate symmetry in strict sense

B.4.2. Phase distance

Let $\widehat{\theta}_k^{(n)}$ and $\theta_k^{(n)}$ to be the phases of $\widehat{y}_k^{(n)}$ and $y_k^{(n)}$, respectively. Then, the phase distance is defined as

$$\mathcal{L}_p = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\left|1 - \exp(j(\widehat{\theta}_k^{(n)} - \theta_k^{(n)}))\right|^2 \tag{27}$$

$$= \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\left|1 - \cos(\widehat{\theta}_k^{(n)} - \theta_k^{(n)})) - j\sin(\widehat{\theta}_k^{(n)} - \theta_k^{(n)}))\right|^2 \tag{28}$$

$$= \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\left[\left(1 - \cos(\widehat{\theta}_k^{(n)} - \theta_k^{(n)}))\right)^2 + \sin(\widehat{\theta}_k^{(n)} - \theta_k^{(n)}))^2\right] \tag{29}$$

$$= \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\left[1 + \cos(\widehat{\theta}_k^{(n)} - \theta_k^{(n)}))^2 + \sin(\widehat{\theta}_k^{(n)} - \theta_k^{(n)}))^2 - 2\cos(\widehat{\theta}_k^{(n)} - \theta_k^{(n)}))\right] \tag{30}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K}\left(1 - \cos(\widehat{\theta}_k^{(n)} - \theta_k^{(n)}))\right) \tag{31}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K}\left[1 - \left(\cos(\widehat{\theta}_k^{(n)})\cos(\theta_k^{(n)})) + \sin(\widehat{\theta}_k^{(n)})\sin(\theta_k^{(n)}))\right)\right] \tag{32}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K}\left[1 - \frac{\mathtt{Re}(\widehat{y}_k^{(n)})\mathtt{Re}(y_k^{(n)}) + \mathtt{Im}(\widehat{y}_k^{(n)})\mathtt{Im}(y_k^{(n)})}{\sqrt{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2}\sqrt{\mathtt{Re}(y_k^{(n)})^2 + \mathtt{Im}(y_k^{(n)})^2}}\right], \tag{33}$$

where

$$\cos(\widehat{\theta}_k^{(n)}) = \frac{\mathtt{Re}(\widehat{y}_k^{(n)})}{\sqrt{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2}}, \quad \cos(\theta_k^{(n)}) = \frac{\mathtt{Re}(y_k^{(n)})}{\sqrt{\mathtt{Re}(y_k^{(n)})^2 + \mathtt{Im}(y_k^{(n)})^2}} \tag{34}$$

$$\sin(\widehat{\theta}_k^{(n)}) = \frac{\mathtt{Im}(\widehat{y}_k^{(n)})}{\sqrt{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2}}, \quad \sin(\theta_k^{(n)}) = \frac{\mathtt{Im}(y_k^{(n)})}{\sqrt{\mathtt{Re}(y_k^{(n)})^2 + \mathtt{Im}(y_k^{(n)})^2}}. \tag{35}$$

Therefore, we get

$$\frac{\partial \mathcal{L}_p}{\partial \mathtt{Re}(\widehat{y}_k^{(n)})} = -\cos(\theta_k^{(n)})\frac{\partial \cos(\widehat{\theta}_k^{(n)})}{\partial \mathtt{Re}(\widehat{y}_k^{(n)})} - \sin(\theta_k^{(n)})\frac{\partial \sin(\widehat{\theta}_k^{(n)})}{\partial \mathtt{Re}(\widehat{y}_k^{(n)})} \tag{36}$$

$$= -\cos(\theta_k^{(n)})\frac{\sqrt{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2} - \mathtt{Re}(\widehat{y}_k^{(n)})\frac{1}{2}\frac{2\mathtt{Re}(\widehat{y}_k^{(n)})}{\sqrt{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2}}}{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2} - \sin(\theta_k^{(n)})\frac{-\mathtt{Im}(\widehat{y}_k^{(n)})\frac{1}{2}\frac{2\mathtt{Re}(\widehat{y}_k^{(n)})}{\sqrt{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2}}}{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2} \tag{37}$$

$$= -\cos(\theta_k^{(n)})\frac{\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2 - \mathtt{Re}(\widehat{y}_k^{(n)})^2}{\left(\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2\right)^{\frac{3}{2}}} - \sin(\theta_k^{(n)})\frac{-\mathtt{Im}(\widehat{y}_k^{(n)})\mathtt{Re}(\widehat{y}_k^{(n)})}{\left(\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2\right)^{\frac{3}{2}}} \tag{38}$$

$$= -\frac{\mathtt{Re}(y_k^{(n)})\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Re}(y_k^{(n)})\mathtt{Im}(\widehat{y}_k^{(n)})^2 - \mathtt{Re}(y_k^{(n)})\mathtt{Re}(\widehat{y}_k^{(n)})^2 - \mathtt{Im}(y_k^{(n)})\mathtt{Im}(\widehat{y}_k^{(n)})\mathtt{Re}(\widehat{y}_k^{(n)})}{\left(\mathtt{Re}(y_k^{(n)})^2 + \mathtt{Im}(y_k^{(n)})^2\right)^{\frac{1}{2}}\left(\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2\right)^{\frac{3}{2}}} \tag{39}$$

$$= -\frac{\mathtt{Re}(y_k^{(n)})\mathtt{Im}(\widehat{y}_k^{(n)}) - \mathtt{Im}(y_k^{(n)})\mathtt{Re}(\widehat{y}_k^{(n)})}{\left(\mathtt{Re}(y_k^{(n)})^2 + \mathtt{Im}(y_k^{(n)})^2\right)^{\frac{1}{2}}\left(\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2\right)^{\frac{3}{2}}}\mathtt{Im}(\widehat{y}_k^{(n)}) \tag{40}$$

$$\frac{\partial \mathcal{L}_p}{\partial \mathtt{Im}(\widehat{y}_k^{(n)})} = -\cos(\theta_k^{(n)})\frac{\partial \cos(\widehat{\theta}_k^{(n)})}{\partial \mathtt{Im}(\widehat{y}_k^{(n)})} - \sin(\theta_k^{(n)})\frac{\partial \sin(\widehat{\theta}_k^{(n)})}{\partial \mathtt{Im}(\widehat{y}_k^{(n)})} \tag{41}$$

$$= -\frac{\mathtt{Im}(y_k^{(n)})\mathtt{Re}(\widehat{y}_k^{(n)}) - \mathtt{Re}(y_k^{(n)})\mathtt{Im}(\widehat{y}_k^{(n)})}{\left(\mathtt{Re}(y_k^{(n)})^2 + \mathtt{Im}(y_k^{(n)})^2\right)^{\frac{1}{2}}\left(\mathtt{Re}(\widehat{y}_k^{(n)})^2 + \mathtt{Im}(\widehat{y}_k^{(n)})^2\right)^{\frac{3}{2}}}\mathtt{Re}(\widehat{y}_k^{(n)}). \tag{42}$$

Because both $\boldsymbol{y}^{(n)}$ and $\widehat{\boldsymbol{y}}^{(n)}$ are conjugate-symmetric, it can be easily observed that $\frac{\partial \mathcal{L}_p}{\partial \mathtt{Re}(\widehat{y}_k^{(n)})}$ and $\frac{\partial \mathcal{L}_p}{\partial \mathtt{Re}(\widehat{y}_k^{(n)})}$ satisfy the condition in Equations (21) and (22), respectively.

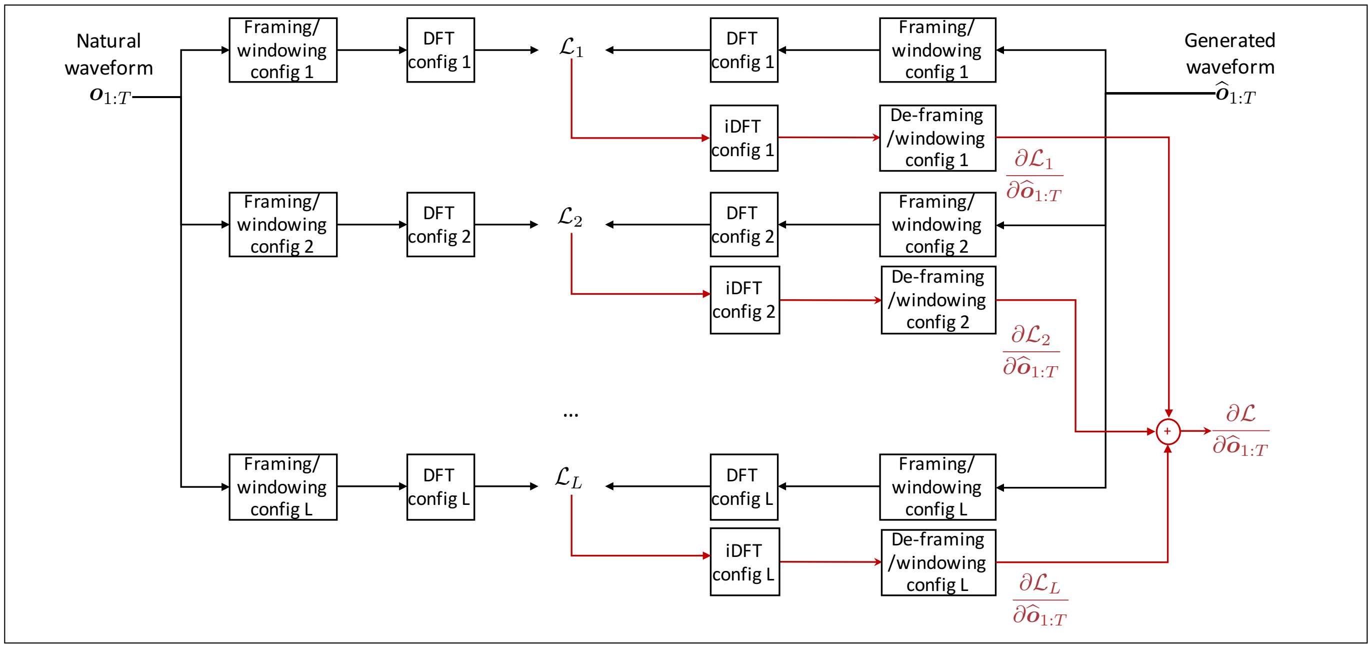
C. MULTIPLE DISTANCE METRICS

Different distance metrics can be merged easily. For example, we can define

$$\mathcal{L} = \mathcal{L}_{s1} + \cdots + \mathcal{L}_{sS} + \mathcal{L}_{p1} + \cdots + \mathcal{L}_{pP}, \tag{43}$$

where $\mathcal{L}_{s*} \in \mathbb{R}$ and $\mathcal{L}_{p*} \in \mathbb{R}$ may use different numbers of DFT bins, frame length, or frame shift. Although the dimension of the gradient vector $\frac{\partial \mathcal{L}_*}{\partial \widehat{\boldsymbol{x}}^{(n)}}$ may be different, the gradient $\frac{\partial \mathcal{L}_*}{\partial \widehat{\boldsymbol{o}}_{1:T}} \in \mathbb{R}^T$ will always be a real-valued vector of dimension T after de-framing/windowing. The gradients then will be simply merged together as

$$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{o}}_{1:T}} = \frac{\partial \mathcal{L}_{s1}}{\partial \widehat{\boldsymbol{o}}_{1:T}} + \cdots + \frac{\partial \mathcal{L}_{sS}}{\partial \widehat{\boldsymbol{o}}_{1:T}} + \frac{\partial \mathcal{L}_{p1}}{\partial \widehat{\boldsymbol{o}}_{1:T}} + \cdots + \frac{\partial \mathcal{L}_{pP}}{\partial \widehat{\boldsymbol{o}}_{1:T}}. \tag{44}$$



Natural waveform $\boldsymbol{o}_{1:T}$
Framing/ windowing config 1
DFT config 1
$\mathcal{L}_1$
DFT config 1
Framing/ windowing config 1
Generated waveform $\widehat{\boldsymbol{o}}_{1:T}$
iDFT config 1
De-framing /windowing config 1
$\frac{\partial \mathcal{L}_1}{\partial \widehat{\boldsymbol{o}}_{1:T}}$
Framing/ windowing config 2
DFT config 2
$\mathcal{L}_2$
DFT config 2
Framing/ windowing config 2
iDFT config 2
De-framing /windowing config 2
$\frac{\partial \mathcal{L}_2}{\partial \widehat{\boldsymbol{o}}_{1:T}}$
...
$\frac{\partial \mathcal{L}}{\partial \widehat{\boldsymbol{o}}_{1:T}}$
Framing/ windowing config L
DFT config L
$\mathcal{L}_L$
DFT config L
Framing/ windowing config L
iDFT config L
De-framing /windowing config L
$\frac{\partial \mathcal{L}_L}{\partial \widehat{\boldsymbol{o}}_{1:T}}$

Fig. 8. Using multiple distances $\{\mathcal{L}_1, \cdots, \mathcal{L}_L\}$