APSIPA Transactions on Signal and Information Processing, 2024, 13, e28

Original Paper

Neural Analog Filter for Sampling-Frequency-Independent Convolutional Layer

Kanami Imamura[1,2*], Tomohiko Nakamura[2], Kohei Yatabe[3] and Hiroshi Saruwatari[1]

[1] *Graduate School of Information Science and Technology, The University of Tokyo, Japan*
[2] *The National Institute of Advanced Industrial Science and Technology (AIST), Japan*
[3] *Department of Electrical Engineering and Computer Science, Tokyo University of Agriculture and Technology, Japan*

ABSTRACT

A sampling-frequency-independent (SFI) convolutional layer is an extension of a convolutional layer to handle various sampling frequencies (SFs) with a single deep neural network (DNN). The SFI convolutional layer treats a standard convolutional layer as a collection of digital filters designed from analog filters. Analogous to the analog-to-digital filter conversion, it generates the weights from an SFI structure (latent analog filter) with respect to an input SF. To use the SFI layer, we need to define the mathematical forms of the latent analog filters before training. However, it is difficult to manually define the appropriate forms for an arbitrary network architecture. The inappropriate definition can result in the performance degradation. To overcome this problem, we propose a neural representation of analog filters, which can determine the forms of the latent analog filters in an end-to-end manner.

*Corresponding author: Kanami Imamura, kanami_imamura@ieee.org. This work was supported by JSPS KAKENHI under Grant JP23K28108 and JST ACT-X under Grant JPMJAX210G.

Received 29 December 2023; revised 17 May 2024; accepted 07 September 2024
ISSN 2048-7703; DOI 10.1561/116.20230082

The proposed method treats the latent analog filter as a function of continuous time or frequency and represents it using a DNN. Music source separation and speech separation experiments showed that the proposed method outperformed manually designed latent analog filters.

Keywords: Audio source separation, sampling-frequency-independent convolutional layer, fourier feature mapping

1 Introduction

Audio source separation is a technique of extracting concurrent sources from a mixture of audio signals. It can be used for the preprocessing of various audio signal processing tasks. To realize an audio source separation method that can be universally used for any downstream tasks, it should handle various recording conditions specified by the tasks. The universality with respect to such recording conditions is essential for constructing a widely applicable audio source separation method.

Sampling frequency (SF) is an essential element in audio signal processing. It is determined by the purpose of a target task, e.g., 44.1 and 48 kHz for music source separation [17] and 8 and 22.05 kHz for automatic speech recognition and automatic music transcription [6, 19, 21, 28]. Despite the variety in SF, audio source separation methods based on deep neural networks (DNNs) generally assume the SF to be the same in the training and inference stages [3, 7, 10, 13, 14, 16, 17, 18, 25, 27, 31, 33, 36, 38]. To handle untrained SFs, we typically resample an input signal to the trained SF before it is fed into the DNN. However, we have experimentally shown that the performance of DNN-based music source separation methods significantly degrades with untrained SFs when using signal resampling [26]. Thus, an alternative approach to signal resampling should be developed to handle untrained SFs for DNN-based audio source separation methods.

As a pioneer of this approach, we previously proposed an SF-independent (SFI) convolutional layer [26]. The SFI layer treats the weights of a usual convolutional layer as the discrete-time impulse responses designed from analog filters. Since the analog filters are defined in the continuous-time or continuous-frequency domain, they do not depend on the SF and can serve as an archetype of the weights for any SFs. Thus, the SFI layer can generate the consistent weights for various SFs by utilizing the analog-to-digital filter conversion methods. We call the archetype the latent analog filter. The SFI layer is replaceable with a usual convolutional layer in a conventional DNN such as Conv-TasNet [14], one of the foundational DNN-based audio source separation

models. Owing to these characteristics, it paved the way for realizing an audio source separation model that can be universally used for various downstream tasks.

To use the SFI layer, we need to define the explicit mathematical forms of the latent analog filters. In our previous paper [26], we used analytic analog filters such as multiphase gammatone filters [5] and modulated Gaussian filters (MGFs). They were derived from the fact that part of Conv-TasNet can be associated with a trainable filterbank. However, such an association is difficult to find for an arbitrary layer. The inappropriate design of the latent analog filters may limit the separation performance. Thus, we need to develop a method that can implicitly represent latent analog filters.

In this paper, we propose a neural representation method for the latent analog filters, *neural analog filter (NAF) representation*, which can determine the latent analog filters in an end-to-end manner. It comes from our finding that an analog filter can be viewed as a function of continuous time or frequency. The proposed method represents this function with a DNN, i.e., we interpret the outputs of the DNN as the values of the impulse or frequency responses at that time or frequency, respectively. We call this DNN *the NAF*. By training the proposed NAFs jointly with the other DNN components, we can determine the latent analog filters without explicitly defining their forms. We applied the proposed NAFs to an SFI extension of a foundational DNN for audio source separation (SFI Conv-TasNet) [26] and examined the effectiveness of the proposed NAFs through music source separation and speech separation experiments.

2 Related Works

2.1 DNN-Based Audio Source Separation

DNN-based audio source separation methods can be classified into two types. One approach separates the spectrogram of the input mixture signal, typically obtained by the short-time Fourier transform (STFT) [7, 16, 33]. In this spectrogram-based approach, the spectrogram of the input mixture signal is fed into a DNN to predict time-frequency masks for individual sources. We call the DNN the mask predictor. The predicted mask of each source is multiplied by the input spectrogram and is converted into the separated signal by the inverse STFT.

The other approach separates the input mixture signal in the waveform domain [10, 13, 14, 18, 25, 27, 36, 38]. It is roughly classified into two types of network architectures. The first type is based on the adaptation of a U-Net architecture for the waveform domain [4, 18, 30]. The second type is constructed by extending a time-frequency transform to be trainable in the

architecture of the spectrogram-based approach, which is called the TasNet framework [15]. In this framework, the time-frequency transform and its inverse are implemented as a convolutional layer (frequently followed by a nonlinear activation function) and a transposed convolutional layer, respectively. The parameters of these layers are trained simultaneously with those of the mask predictor. This framework is adopted in most methods of this approach, which focuses on the improvement of the architecture of the mask predictor [10, 14, 27, 36, 38].

Some recently proposed methods utilize both spectrogram-based and waveform-based approaches [3, 25]. These models consist of a waveform-domain network, a spectrogram-domain network, and a network that bridges the two domains. The bridging network integrates the features in the bottleneck layers of the waveform- and spectrogram-domain networks and propagates the information extracted from the two domains to the subsequent layers.

2.2 Methods for Handling Untrained SFs

A few methods have been proposed to handle untrained SFs as an alternative to the signal resampling [20, 26]. The SFI convolutional layer is the first layer for handling untrained SFs [26]. The key of this layer is to generate the weights of a usual convolutional layer from the latent analog filters, which are defined in the continuous-time or frequency domain, i.e., the SFI domain. The parameters of the latent analog filters are also defined in the SFI domain; thus, the SFI convolutional layer can handle various SFs by generating the weights in accordance with the input SF. We will describe the details of this generation process later in Section 2.3. The advantage of this layer is that it can be replaced with a usual convolutional layer because their architectural difference is only the weight generation. Thus, we can easily introduce the SFI convolutional layer into various DNNs using a convolutional layer, for example, Conv-TasNet [14].

The other method can handle untrained SFs by restricting network architectures [20]. It targets a spectrogram-based architecture that is agnostic to time-frequency axes and does not contain pooling and unpooling layers, e.g., a two-dimensional convolutional neural network with strides of one. Paulus *et al.* demonstrated that this architecture works well with untrained SFs by adjusting the window length and hopsize of the STFT to keep the time resolution of the input spectrogram in the unit of seconds. Although it has an advantage in simplicity, the architectural requirements are too restrictive in modern DNNs for audio source separation. For example, MMDenseNet [34], one of the spectrogram-based DNNs, violates these requirements because it splits the frequency band and separates the input mixture signal based on local patterns of each frequency band. In terms of the applicability to various network architectures, the SFI convolutional layer has an advantage and we target it throughout this paper.

2.3 SFI Convolutional Layer

We briefly describe an SFI convolutional layer with C input channels and C' output channels. Let $c = 0, \ldots, C-1$ and $c' = 0, \ldots, C'-1$ be the input and output channel indices, respectively. This layer consists of $C \times C'$ latent analog filters and a usual convolutional layer. Let t be the continuous time and ω be the unnormalized angular frequency. Each latent analog filter is defined as a real-valued continuous impulse response $g_{c',c}(t)$ or a complex-valued continuous frequency response $G_{c',c}(\omega)$.

Given a target SF $F_s^{(\mathrm{target})}$, the SFI convolutional layer generates a discrete-time impulse response $\{b_{c',c}[n]\}_{n=\lfloor -(N-1)/2 \rfloor}^{\lfloor (N-1)/2 \rfloor}$ from each latent analog filter using a digital filter design technique, where n is the discrete-time index and N is the kernel size. Since a usual convolutional layer computes a cross-correlation between an input and weights, $b_{c',c}[n]$ is flipped along the time axis and is used as the weights. We obtain an output by decimating the cross-correlation by a factor stride S. We can also define an SFI version of a usual transposed convolutional layer (the SFI transposed convolutional layer) by replacing the convolutional layer with the transposed convolutional layer.

To generate $b_{c',c}[n]$, we can choose the time-domain (TD) or frequency-domain (FD) filter design methods. To distinguish the SFI layers with the two methods, we call the former *the TD SFI layer* and the latter *the FD SFI layer*.

TD filter design: The TD filter design method directly samples $g_{c',c}(t)$ with an interval of $1/F_s^{(\mathrm{target})}$:

$$b_{c',c}[n] = g_{c',c}(n/F_s^{(\mathrm{target})}). \tag{1}$$

Owing to the Nyquist theorem, the direct sampling causes aliasing when $g_{c',c}(t)$ has energies in the frequency band above the Nyquist frequency $F_s^{(\mathrm{target})}/2$.

To avoid the aliasing, we previously developed an aliasing reduction method based on the center frequency of $g_{c',c}(t)$ [26]. It forcibly sets $b_{c',c}[n]$ to zero for all n when the center frequency of $g_{c',c}(t)$ is above the target Nyquist frequency $F_s^{(\mathrm{target})}/2$. This method is effective when the energies of $g_{c',c}(t)$ are distributed around the center frequency.

FD filter design: The FD filter design method approximates $G_{c',c}(\omega)$ using the frequency response of the digital filter in the least-squares sense. Let $\{\omega_k\}_{k=0}^{K-1}$ be the series of K angular frequencies that satisfy $\omega_1 < \omega_2 < \ldots < \omega_K$ and $\omega_k \in [0, F_s^{(\mathrm{target})}/2]$ for all k. The discrete-time impulse response is given as a solution of the following problem:

$$\boldsymbol{b}_{c',c} = \underset{\tilde{\boldsymbol{b}} \in \mathbb{R}^N}{\arg\min} \|\boldsymbol{G}_{c',c} - D\tilde{\boldsymbol{b}}\|_2^2, \tag{2}$$

where $\boldsymbol{b}_{c',c}$ and $\boldsymbol{G}_{c',c}$ are the vectorized forms of $\{b_{c',c}[n]\}_{n=\lfloor -(N-1)/2 \rfloor}^{\lfloor (N-1)/2 \rfloor}$ and $\{G_{c',c}(\omega_k)\}_{k=0}^{K-1}$, respectively. The second term of (2) corresponds to the

discrete-time Fourier transform of $\tilde{b}$ sampled at $\{\omega_k\}_{k=0}^{K-1}$, and $D \in \mathbb{C}^{K\times N}$ is a matrix whose (k,n)th entry is given by $\exp[j\omega_k(n-N/2)/F_s^{(\mathrm{target})}]$. This problem can be solved analytically as

$$\boldsymbol{b}_{c',c} = \begin{bmatrix} \Re D \\ \Im D \end{bmatrix}^{\dagger} \begin{bmatrix} \Re \boldsymbol{G}_{c',c} \\ \Im \boldsymbol{G}_{c',c} \end{bmatrix}, \tag{3}$$

where $\Re$ and $\Im$ return the real and imaginary parts of each matrix and vector entry, respectively. The superscript $\dagger$ denotes the Moore–Penrose pseudo inverse of a matrix. The FD method explicitly uses the Nyquist frequency and the aliasing reduction is introduced by design.

3 Proposed NAF Representation

In this section, we propose two NAFs for the TD and FD filter design methods, respectively. To reduce the number of the parameters, we designed the NAFs to output jointly the values of the impulse or frequency responses of all channels at the input time or frequency, respectively.

3.1 DNN for Latent Analog Filter Representation

A naive choice for the NAFs is a multilayer perceptron (MLP) because it can approximate various functions accurately when using enough units and appropriate nonlinearities [29]. However, using a simple MLP for the NAFs greatly degraded the separation performance in a preliminary experiment. To examine the cause of this degradation, we analyzed the magnitude frequency responses of the trained NAFs. It turned out that most of their energies were distributed in the low frequency band, as we will show later in Section 4.2.

To reduce this bias, we propose using an input transformation method, the Fourier feature mapping (FFM) [35]. It is inspired from our finding that the observed distribution bias in energy resembles the phenomenon (spectral bias) that an MLP prioritizes training the low-frequency components of a target function [1, 24]. The spectral bias can be reduced by introducing the FFM for MLP-based image synthesis [35]. Thus, we decided to adopt the FFM for the proposed NAFs.

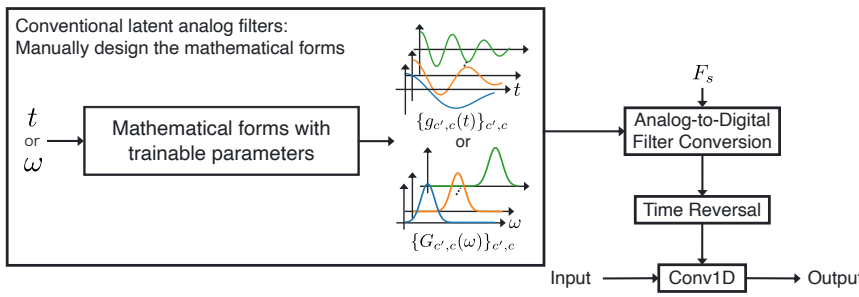
The FFM $\boldsymbol{\gamma}(\cdot)$ transforms a given input $x \in \mathbb{R}$ to a $2R$-dimensional vector using R parameters $\{v_r\}_{r=1}^{R}$:

$$(\boldsymbol{\gamma}(x))_r = \begin{cases} \cos(2\pi v_r x) & (r = 1, \ldots, R) \\ \sin(2\pi v_{r-R} x) & (r = R+1, \ldots, 2R) \end{cases}, \tag{4}$$

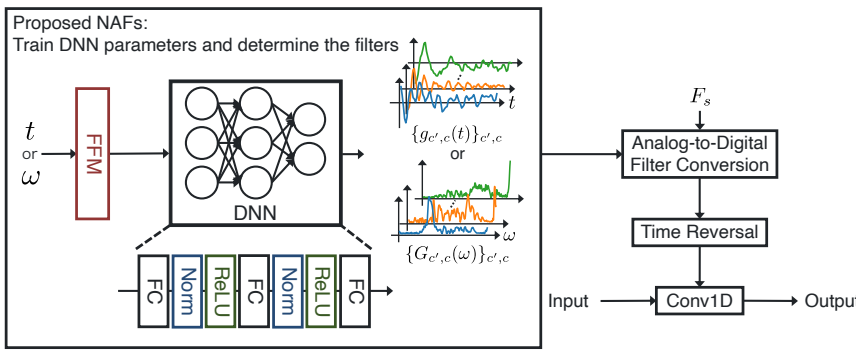
where $(\boldsymbol{\gamma}(x))_r$ is the r-th element of $\boldsymbol{\gamma}(x)$. The parameter v_r is usually determined by random sampling from a standard Gaussian distribution, but it can be trained jointly with the other DNN parameters.

3.2 TD SFI Layer Using NAFs

Figure 1 shows the architecture of the TD and FD SFI layers using the proposed NAFs. The NAFs transform t or ω with the FFM and pass $\boldsymbol{\gamma}(t)$ or $\boldsymbol{\gamma}(\omega)$ to an MLP, respectively. The MLP outputs are used for the weight generation. Owing to the high expressive power of MLPs, we can determine the latent analog filters more flexibly compared with the manually designed filters.



(a) Conventional manually designed filters.



(b) Proposed NAFs.

Figure 1: Architecture of SFI convolutional layer using conventional filters and proposed NAFs. "FC", "Norm", and "ReLU" denote fully-connected layer, layer normalization, and rectified linear unit activation, respectively.

For the TD SFI layer, we need to reduce aliasing caused by sampling from the latent analog filters, as described in Section 2.3. The aliasing reduction method proposed by Saito *et al.* [26] assumes that the energies of $g_{c',c}(t)$ are distributed around the center frequency. However, this assumption does not always hold for the NAFs because $g_{c',c}(t)$ is implicitly defined.

To solve this problem, we propose an anti-aliasing method using oversampling (see Figure 2). The oversampling is the process of sampling a continuous-time signal with an SF that is sufficiently higher than the target SF. The

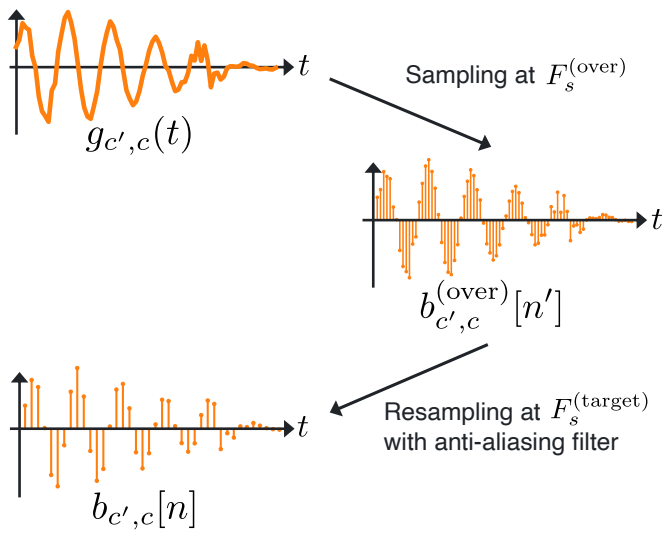


Figure 2: Illustration of proposed anti-aliasing method based on oversampling.

proposed method applies it to the NAFs and generates the $F_s^{(\mathrm{over})}$ Hz-sampled discrete-time impulse responses of length N', $\{b_{c',c}^{(\mathrm{over})}[n']\}_{n'=\lfloor -(N'-1)/2 \rfloor}^{\lfloor (N'-1)/2 \rfloor}$, where $N' = F_s^{(\mathrm{over})} N / F_s^{(\mathrm{target})}$. Resampling $b_{c',c}^{(\mathrm{over})}[n']$ (with an anti-aliasing filter) yields the discrete-time impulse responses $b_{c',c}[n]$ of the target SF $F_s^{(\mathrm{target})}$. While the NAFs are not guaranteed to have zero energies in the frequency above $F_s^{(\mathrm{over})}/2$ Hz, we empirically found that the proposed method works well by setting $F_s^{(\mathrm{over})}$ to the trained SF.

3.3 FD SFI Layer Using NAFs

The FD SFI layer using the proposed NAFs does not require an anti-aliasing method, but it requires additional care in handling SFs higher than the trained SF. For these SFs, the range used in the filter design contains the frequencies higher than the trained Nyquist frequency. The proposed NAFs should not provide proper outputs for the untrained frequencies. In a preliminary experiment, we observed that using the NAF outputs above the trained Nyquist frequency degraded the separation performance. Thus, we apply the ideal low-pass filter with a cutoff frequency of the trained Nyquist frequency to $\boldsymbol{G}_{c',c}$. That is, we forcibly set $G_{c',c}(\omega) = 0$ for ω above the trained Nyquist frequency.

3.4 Comparison of TD and FD SFI Layers Using NAFs

The main difference between the TD and FD SFI layers using the proposed NAFs is the treatment of aliasing, as described in Sections 3.2 and 3.3. The TD SFI layer generates the weights by the direct sampling from the latent analog filters and is easy to implement. However, it requires the additional oversampling-based anti-aliasing method with an adequate $F_s^{(\mathrm{over})}$. The FD SFI layer can avoid the aliasing by design, but its implementation is less simple than that of the TD SFI layer. Thus, the decision as to whether to use the TD and FD SFI layers depends on whether we prioritize anti-aliasing or ease of implementation.

4 Experimental Evaluation in Music Source Separation

4.1 Experimental Settings

Data: To evaluate the proposed NAF representation, we conducted music source separation experiments using the MUSDB18-HQ dataset [23]. This dataset contains 150 tracks and each track contains audio signals of four musical instruments (*vocals*, *bass*, *drums*, and *other*). We used the official data split provided in the `musdb` library (https://github.com/sigsep/sigsep-mus-db): 86, 14, and 50 tracks for training, validation, and test respectively. The trained SF $F_s^{(\mathrm{train})}$ was set to 32 kHz. The test data were created by resampling the test tracks at SFs from 8 to 48 kHz with an interval of 4 kHz.

Evaluation metric: The evaluation metric was the signal-to-distortion ratio (SDR) calculated by the BSSEval v4 toolkit [32]. To reduce the dependency on the initial values, we computed the SDRs for models with four random seeds, and their means and standard errors were used for comparison.

Network architecture: All the methods used the network architecture of SFI Conv-TasNet [26]. The SFI Conv-TasNet architecture is an SFI extension of the Conv-TasNet architecture defined in Samuel *et al.* [27].[1] It consists of an encoder, four mask predictors, and a decoder. The encoder transforms an input signal into a pseudo time-frequency representation using the SFI convolutional layer and ReLU. The mask predictors predict masks from the encoder output for individual sources. The decoder converts the masked pseudo time-frequency representations into separated signals using the SFI transposed convolutional layer. Table 1 shows the hyperparameters of SFI Conv-TasNet. We set the same hyperparameters as Saito *et al.* [26].

[1] In Samuel *et al.* [27], the Conv-TasNet was adapted for music source separation and modified from the original Conv-TasNet architecture [14]. See Saito *et al.* [26, Section V-A] for its details.

Table 1: Hyperparameters of SFI Conv-TasNet for music source separation

Hyperparameter	Value
Number of output channel of encoder and input channel of decoder	440
Number of channels in bottleneck and residual paths' 1×1 convolutional blocks	160
Number of channels in skip-connection paths' 1×1 convolutional blocks	160
Number of channels in convolutional blocks	160
Kernel size in convolutional blocks	3
Number of convolutional blocks in each repeat	6
Number of repeats of convolutional blocks	2
N of encoder and decoder in training	160
S of encoder and decoder in training	80

In inference, the time resolution of the encoder output should be constant regardless of SFs since the mask predictors do not include SFI convolutional layers. Therefore, we changed the kernel size and stride so that these values were the same in the unit of seconds during training and inference, as in Saito *et al.* [26]. For example, when we set $N = 160$ and $S = 80$ (5 ms and 2.5 ms, respectively) at the SF of 32 kHz, we use $N = 80$ and $S = 40$ (5 ms and 2.5 ms, respectively) at the SF of 16 kHz.

Training configurations: We used the same training configurations as in Saito *et al.* [26]. The models were trained for 250 epochs using RAdam [12] and LookAhead [39]. The loss function was the negative scale-invariant source-to-noise ratio (SI-SNR) [9]. SI-SNR between the estimated signals $\hat{a}[n]$ and ground-truth signals $a[n]$ is denoted as

$$a_{\text{target}}[n] = \frac{\sum_{n'}(\hat{a}[n']a[n'])}{\sum_{n'}(a[n'])^2}a[n], \tag{5}$$

$$e_{\text{noise}}[n] = \hat{a}[n] - a_{\text{target}}[n], \tag{6}$$

$$\text{SI-SNR} = 10\log_{10}\frac{\sum_n(a_{\text{target}}[n])^2}{\sum_n(e_{\text{noise}}[n])^2}. \tag{7}$$

Owing to the scale invariance of the loss function, the outputs of the trained models may have different scales from the groundtruth signals. To compensate for the scale ambiguity, we adjusted the scales of the network outputs to align with the input mixture in scale, as in Samuel *et al.* [27], which is denoted as

$$\{\alpha_m\}_{m=1}^M = \arg\min_{\{\tilde{\alpha}_m\}_{m=1}^M} \sum_{n=0}^{N-1}\left(y[n] - \sum_{m=1}^M \tilde{\alpha}_m\hat{a}_m[n]\right)^2, \tag{8}$$

where $y[n]$ is the mixture signal, $\hat{a}_m[n]$ is the estimated signal, and α_m is the scale for source m.

Methods: Table 2 shows the methods used in the experiments, where TD and FD mean the use of the TD and FD filter design methods, respectively. TD- and FD-MGFs used the manually designed latent analog filters, MGFs, which achieved the highest separation performance on average in Saito *et al.* [26]. The impulse response of the MGF is given as

$$g^{(\mathrm{MGF})}(t) = 2\sqrt{2\sigma^2\pi}\exp\left(-\frac{\sigma^2 t^2}{2}\right)\cos(\mu t + \phi), \tag{9}$$

where μ is the center frequency in radians, σ is a parameter corresponding to the bandwidth, and ϕ is the phase shift. These parameters were defined for each pair of input and output channels. σ^2 was initialized with $(80\pi)^2$, and the other parameters were the same as those in Saito *et al.* [26].

Table 2: Characteristics of compared methods

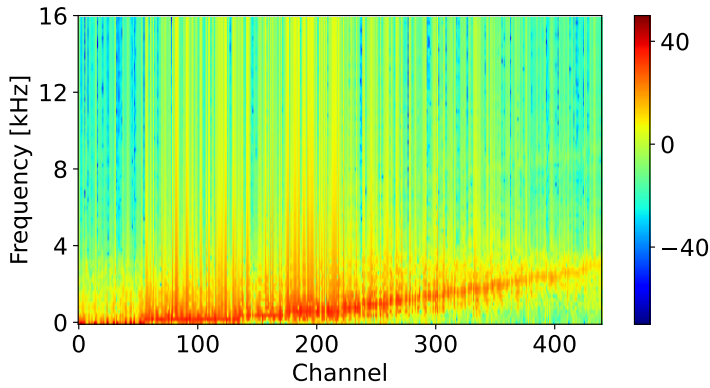
Method	Latent analog filter	Domain of filter design	Anti-aliasing method
TD-MGF	$g^{(\mathrm{MGF})}(t)$	Time	[26]
FD-MGF		Frequency	-
TD-NAF	**Proposed NAF**	Time	**Oversampling-based**
FD-NAF		Frequency	-

TD-NAF and FD-NAF used the proposed NAFs instead of the MGFs. TD-NAF used the oversampling-based anti-aliasing method proposed in Section 3.2. For a fair comparison, all the hyperparameters except for the NAF parameters were the same as those for TD-MGF and FD-MGF.

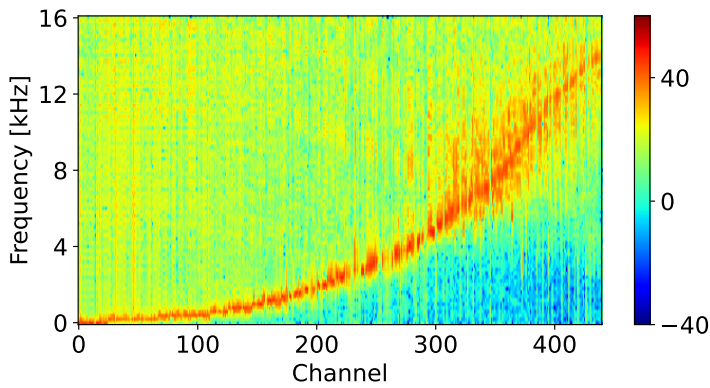
Figure 1 shows the network architecture for the NAFs. The first FC layer had 256 input units (i.e., $R = 128$). The last FC layer had 440 and 880 output channels for TD- and FD-NAFs, respectively. For FD-NAF, the first 440 output channels were for $\Re\boldsymbol{G}_{c',c}$ and the other channels were for $\Im\boldsymbol{G}_{c',c}$. The input and output channels of the other FC layers were set to 224. The FFM inputs t and ω were normalized as $t/F_s^{(\mathrm{target})}$ and $\omega/F_s^{(\mathrm{train})}$ to squash their values within $[0, 1]$, respectively. We initialized v_r with the standard normal distribution and jointly trained it with the other layers. We found that this joint training slightly improved the SDRs in a preliminary experiment.

4.2 Effect of Using FFM

We first examined the effect of the FFM. Figure 3 shows the magnitude frequency responses of the trained TD-NAF encoders with and without the



(a) Without FFM



(b) With FFM

Figure 3: Magnitude frequency responses of TD-NAF encoders.

FFM. The energies of the NAFs without the FFM were distributed in the frequency band below 4 kHz, showing that the high-frequency components tended to be ignored. For FD-NAF, we observed a similar trend for the magnitude frequency responses of the trained encoders. We also found that the SDRs of FD-NAF without the FFM quickly decreased as the SF moved away from the trained SF. For example, it had SDRs of around 0 dB for all instruments at an SF of 8 kHz, although it provided similar SDRs to FD-NAF with the FFM at an SF of 32 kHz. These results clearly demonstrate the effectiveness of using the FFM. We hereafter show the results obtained with the methods using the FFM.

4.3 Effect of Using Proposed Anti-aliasing Method

We next examined the effect of the proposed anti-aliasing method for TD-NAF. Table 3 shows the SDRs of TD-NAFs with and without the proposed anti-aliasing method for 8, 12, and 16 kHz-sampled data. For the other SFs, the two methods had similar SDRs. The lack of the anti-aliasing method caused the SDRs to drop substantially for 8 and 12 kHz, demonstrating the effectiveness of the proposed anti-aliasing method. As the SF increased, the SDR drop became less because the aliasing artifacts decreased.

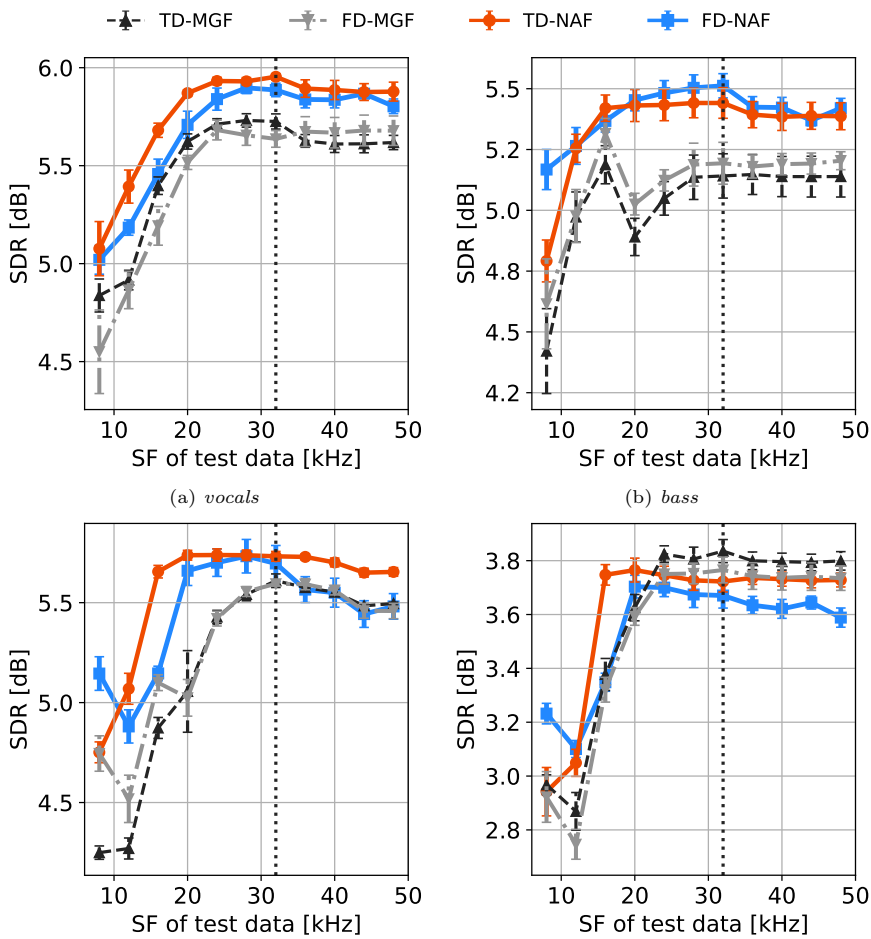
Table 3: SDRs [dB] with and without proposed anti-aliasing method in TD-NAF and p-values of statistical tests

SF	Instrument	Anti-aliasing method		p-value
		Without	With	
8 kHz	*vocals*	1.04 ± 0.35	5.08 ± 0.32	$< 10^{-18}$
	bass	3.40 ± 0.10	4.80 ± 0.20	$< 10^{-4}$
	drums	1.91 ± 0.23	4.75 ± 0.12	$< 10^{-15}$
	other	0.53 ± 0.31	2.94 ± 0.21	$< 10^{-14}$
12 kHz	*vocals*	2.58 ± 0.60	5.39 ± 0.20	$< 10^{-16}$
	bass	4.02 ± 0.22	5.25 ± 0.13	$< 10^{-5}$
	drums	3.21 ± 0.34	5.07 ± 0.18	$< 10^{-14}$
	other	1.82 ± 0.24	3.05 ± 0.12	$< 10^{-9}$
16 kHz	*vocals*	5.26 ± 0.09	5.68 ± 0.08	$< 10^{-8}$
	bass	5.32 ± 0.11	5.42 ± 0.13	0.03
	drums	5.38 ± 0.18	5.66 ± 0.07	$< 10^{-9}$
	other	3.56 ± 0.14	3.75 ± 0.09	$< 10^{-7}$

Table 3 also shows the p-values of the statistical tests for SDRs with and without the proposed anti-aliasing method. We computed the p-values by paired t-tests for the SDRs obtained for the 50 test tracks. TD-NAF with the proposed anti-aliasing method had higher SDRs than without the proposed method at 8, 12, and 16 kHz and for all the instruments with a significance level of 5%.

4.4 Comparison with Manually Designed Latent Analog Filters

We compared TD- and FD-NAFs with TD- and FD-MGFs. Figure 4 shows the SDRs of all the methods per instrument. TD-NAF had higher SDRs than TD-MGF at SFs lower than 24 kHz for all the instruments. For the SFs of 24 kHz and higher, TD-NAF had similar SDRs to TD-MGF for other and higher SDRs than TD-MGF for vocals, bass, and drums. As shown in Table 4, TD-NAF had significantly higher SDRs than TD-MGF at lower SFs for all the instruments. TD-NAF also had significantly higher SDRs at the trained SF for drums and bass. FD-NAF provided comparable or higher SDRs than



(a) *vocals* (b) *bass*

(c) *drums* (d) *other*

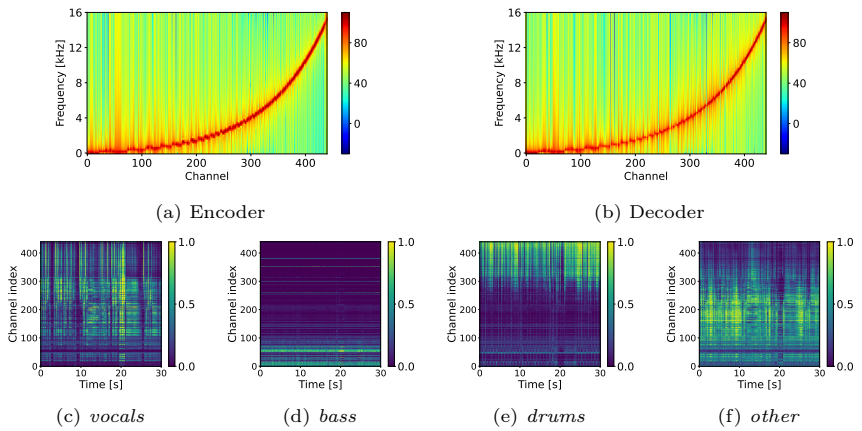
Figure 4: Averages and standard errors of SDRs for SFI Conv-TasNets using MGFs and proposed NAFs. Black dotted lines denote trained SFs.

FD-MGF at SFs lower than the trained SF. For the SFs higher than the trained SF, FD-NAF showed a higher SDR than FD-MGF for vocals and bass and similar SDRs for drums. Although FD-NAF had lower SDRs than FD-MGF for other, their gap was slight (at most 0.15 dB) and typically inaudible. These results show that using the NAFs achieved higher or comparable separation performance compared with the manually designed latent analog filters.

Figure 5 shows the frequency responses of TD-MGF's encoder and decoder, and examples of the predicted masks, and Figure 6 shows those of TD-NAF.

Table 4: p-values of statistical tests for SDRs of TD-MGF and TD-NAF

SF	Instrument			
	vocals	*bass*	*drums*	*other*
8 kHz	0.002	$< 10^{-7}$	$< 10^{-8}$	0.008
16 kHz	$< 10^{-6}$	0.003	$< 10^{-31}$	$< 10^{-18}$
24 kHz	0.02	0.0003	$< 10^{-10}$	0.3
32 kHz	0.4	0.02	$< 10^{-6}$	0.9

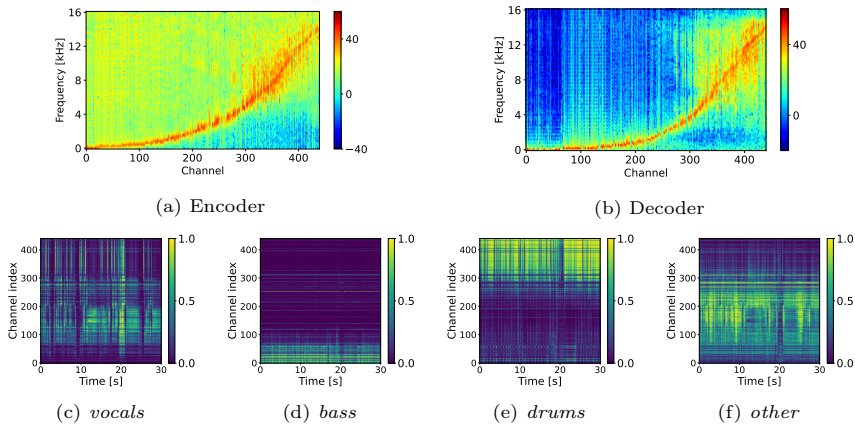


(a) Encoder (b) Decoder

(c) *vocals* (d) *bass* (e) *drums* (f) *other*

Figure 5: Frequency responses of TD-MGF's encoder and decoder, and examples of predicted masks for individual sources at 32 kHz.

For clear visualization, we first calculated the frequency index with the largest magnitude of the frequency response for each channel. We then sorted the channels so that these frequency indexes were in ascending order. Similar to TD-MGF, TD-NAF had frequency responses in which each channel corresponds to a different frequency band, especially in the lower frequency range. TD-MGF and TD-NAF had similar trends in the distribution of center frequencies. For the channels with center frequencies below 6 kHz (the channel indexes below around 350), the frequency responses of TD-NAF and TD-MGF had similar trends. Some channels with the center frequencies above 6 kHz had energies in a broader frequency band. Nevertheless, the predicted masks of TD-NAF were active in fewer channels than those of TD-MGF. It suggests that the TD-NAF encoder encoded sources more compactly without sacrificing the separation performance.



(a) Encoder (b) Decoder

(c) *vocals* (d) *bass* (e) *drums* (f) *other*

Figure 6: Frequency responses of TD-NAF's encoder and decoder, and examples of predicted masks for individual sources at 32 kHz.

4.5 Evaluation on MoisesDB

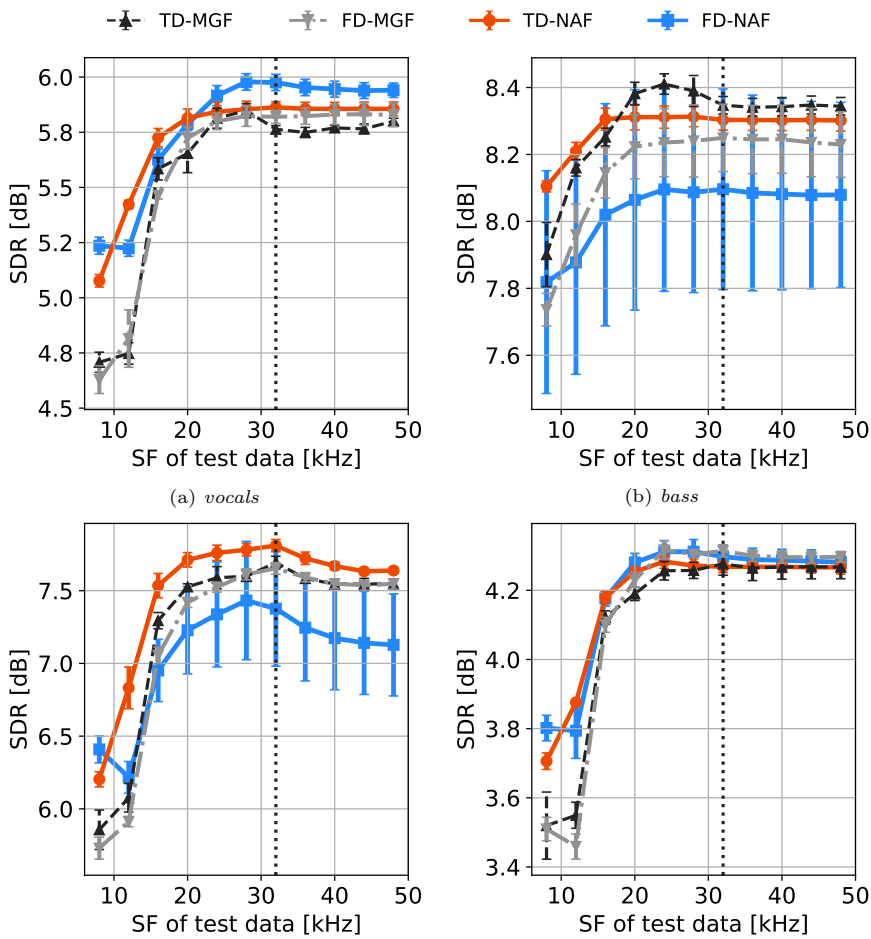
We further compared the NAF and MGF methods by another music source separation dataset, MoisesDB [22]. This dataset contains 240 tracks. We used 235 tracks consisting of four musical instruments: vocals, bass, drums, and other. We split 235 tracks into 132, 25, and 78 tracks for training, validation, and test, respectively. The other settings were the same as in Section 4.1.

Figure 7 shows the SDRs of all the methods per instrument. We observed similar trends to the results of MUSDB18-HQ. Table 5 shows the p-values of the statistical tests for SDRs of TD-MGF and TD-NAF. We computed the p-values by paired t-tests for the SDRs obtained for the 78 test tracks. These results showed that TD-NAF provided higher SDRs than TD-MGF at 8 kHz for all the instruments and at 16, 24, and 32 kHz for vocals with a significance level of 5%. In FD-NAF, the standard errors of SDRs are large for bass and drums, suggesting a high initial value dependence. For vocals and others, the SDRs of FD-NAF were higher than those of FD-MGF at lower SFs. These results provide another evidence that the proposed NAFs are effective for handling untrained SFs in music source separation.

5 Experimental Evaluation in Speech Separation

5.1 Experimental Settings

To examine the generality of the proposed NAFs, we conducted a speech separation experiment.



(a) *vocals* (b) *bass*

(c) *drums* (d) *other*

Figure 7: Averages and standard errors of SDRs of TD-MGF, FD-MGF, TD-NAF, and FD-NAF trained with MoisesDB. Black dotted lines denote trained SFs.

Data: The commonly used WSJ0-2mix dataset [8] and LibriMix [2] are not suitable for this experiment, since SFs of these datasets are 16 kHz. Therefore, we used the VCTK corpus [37], which consists of speech data with SFs of 96 kHz, and created two-speaker mixtures in the same way as the WSJ0-2mix dataset. We divided 109 speakers into 93 speakers for train and validation and 16 speakers for test. We generated the two-speaker mixtures by randomly selecting two utterances from different speakers and mixing them with a random signal-to-noise ratio between 0 and 5 dB. We generated 20000 mixtures for

Table 5: p-values of statistical tests for SDRs of TD-MGF and TD-NAF

SF	Instrument			
	vocals	*bass*	*drums*	*other*
8 kHz	$< 10^{-4}$	$< 10^{-5}$	$< 10^{-10}$	0.03
16 kHz	$< 10^{-9}$	1	$< 10^{-10}$	0.09
24 kHz	$< 10^{-5}$	0.12	0.02	0.2
32 kHz	$< 10^{-5}$	0.2	0.1	0.05

train, 5000 mixtures for validation, and 3000 mixtures for test.

Evaluation metric: The evaluation metric was the SI-SNR improvement (SI-SNRi). SI-SNRi was calculated for each utterance and averaged across the test datasets. To reduce the dependency on the initial values, we computed the SI-SNRi for models with four random seeds, and their means and standard errors were used for comparison.

Network architecture: The speech separation model was based on the original Conv-TasNet [14]. We replaced the convolutional and transposed convolutional layers of the encoder and decoder with the SFI convolutional and SFI transposed convolutional layers, respectively. Table 6 shows the hyperparameters of SFI Conv-TasNet. Since N and S were set to 16 and 8 (2 ms and 1 ms, respectively) at the SF of 8 kHz in Luo and Mesgarani [14], we used $N = 64$ and $S = 32$ (2 ms and 1 ms, respectively) at the trained SF of 32 kHz. The other hyperparameters were the same as Luo and Mesgarani [14].

Table 6: Hyperparameters of SFI Conv-TasNet for speech separation

Hyperparameter	Value
Number of output channel of encoder and input channel of decoder	512
Number of channels in bottleneck and residual paths' 1×1 convolutional blocks	128
Number of channels in skip-connection paths' 1×1 convolutional blocks	128
Number of channels in convolutional blocks	512
Kernel size in convolutional blocks	3
Number of convolutional blocks in each repeat	8
Number of repeats of convolutional blocks	3
N of encoder and decoder in training	64
S of encoder and decoder in training	32

Training configuration: We used the same training configurations as [14]. The models were trained for 100 epochs using Adam [11]. The loss function was the negative SI-SNR.

Methods: We compared four methods in Table 2. Figure 1 shows the network architecture for the NAFs. The last FC layer had 512 and 1024 output channels for TD- and FD-NAFs, respectively. The input and output channels of the other FC layers were set to 64. The other parameters were the same as in Section 4.1.

5.2 Results

Figure 8 shows the SI-SNRi of all the methods. TD-NAF had higher SI-SNRi than TD-MGF at SFs lower than the trained SF and similar SI-SNRi at SFs higher than the trained SF. FD-NAF had higher SI-SNRi than FD-MGF at all SFs. Table 7 shows the p-values of the statistical tests for SI-SNRi of TD-MGF and TD-NAF. We computed the p-values by paired t-tests for the SI-SNRi obtained for the 3000 test utterances. NAFs provided higher SI-SNRi than MGFs at 8, 16, 24, and 32 kHz with a significance level of 5%. These results clearly demonstrate the effectiveness of NAFs in speech separation.

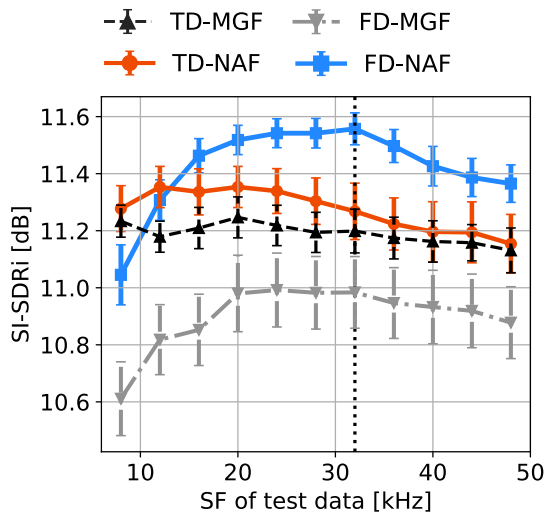


Figure 8: Averages and standard errors of SI-SNRi of TD-MGF, FD-MGF, TD-NAF, and FD-NAF. Black dotted lines denote trained SFs.

Table 7: p-values of statistical tests for SI-SNRi of NAF and MGF models

SF	NAF vs. MGF	
	TD	FD
8 kHz	0.02	$< 10^{-90}$
16 kHz	$< 10^{-9}$	$< 10^{-164}$
24 kHz	$< 10^{-10}$	$< 10^{-147}$
32 kHz	$< 10^{-3}$	$< 10^{-143}$

6 Conclusion

We proposed an NAF representation that enables us to train latent analog filters without explicitly defining their forms. It treats the latent analog filter as a function of continuous time or frequency. By representing this function using a DNN combined with the FFM, we can determine the forms of the latent analog filters by training. We further proposed the oversampling-based anti-aliasing method for the NAFs to reduce the aliasing artifacts caused by the TD filter design method. By applying the proposed NAFs to SFI Conv-TasNet, we demonstrated the effectiveness of the proposed NAFs through music source separation and speech separation experiments.

References

[1] Y. Cao, Z. Fang, Y. Wu, D.-X. Zhou, and Q. Gu, "Towards understanding the spectral bias of deep learning", in *Proceedings of International Joint Conferences on Artificial Intelligence*, 2021, 2205–11.

[2] J. Cosentino, M. Pariente, S. Cornell, A. Deleforge, and E. Vincent, "LibriMix: An open-source dataset for generalizable speech separation", *arxiv:2005.11262*, 2020.

[3] A. Défossez, "Hybrid spectrogram and waveform source separation", in *Proceedings of International Society for Music Information Retrieval Conference*, 2021.

[4] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain", *arXiv:1911.13254*, 2019.

[5] D. Ditter and T. Gerkmann, "A multi-phase gammatone filterbank for speech separation via TasNet", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, 36–40.

[6] J. Gao, J. Du, and E. Chen, "Mixed-bandwidth cross-channel speech recognition via joint optimization of DNN-based bandwidth expansion and acoustic modeling", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(3), 2019, 559–71.

[7] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models", *Journal of Open Source Software*, 5(50), 2020, 2154.

[8] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, 31–5.

[9] Y. Isik, J. Le Roux, Z. Chen, S. Watanabe, and J. Hershey, "Single-channel multi-speaker separation using deep clustering", in *Proceedings of INTERSPEECH*, 2016, 545–9.

[10] I. Kavalerov, S. Wisdom, H. Erdogan, B. Patton, K. Wilson, J. Le Roux, and J. R. Hershey, "Universal sound separation", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, 175–9.

[11] D. Kingma and J. Ba, "Adam: A method for stochastic optimization", in *Proceedings of International Conference on Learning Representations*, 2014.

[12] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond", in *Proceedings of International Conference on Learning Representations*, 2020.

[13] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path RNN: Efficient long sequence modeling for time-domain single-channel speech separation", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, 46–50.

[14] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time-frequency magnitude masking for speech separation", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8), 2019, 1256–66.

[15] Y. Luo and N. Mesgarani, "TasNet: Time-domain audio separation network for real-time, single-channel speech separation", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, 696–700.

[16] Y. Luo and J. Yu, "Music source separation with band-split RNN", *arXiv:2209.15174*, 2022.

[17] Y. Mitsufuji, G. Fabbro, S. Uhlich, F.-R. Stöter, A. Défossez, M. Kim, W. Choi, C.-Y. Yu, and K.-W. Cheuk, "Music demixing challenge 2021", *frontiers*, 1, Article 808395, 2022, 1–14.

[18] T. Nakamura, S. Kozuka, and H. Saruwatari, "Time-domain audio source separation with neural networks based on multiresolution analysis", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, 2021, 1687–701.

[19] A. Narayanan, A. Misra, K. C. Sim, G. Pundak, A. Tripathi, M. Elfeky, P. Haghani, T. Strohman, and M. Bacchiani, "Toward domain-invariant speech recognition via large scale training", in *Proceedings of IEEE Workshop on Spoken Language Technology*, 2018, 441–7.

[20] J. Paulus and M. Torcoli, "Sampling frequency independent dialogue separation", in *Proceedings of European Signal Processing Conference*, 2022, 160–4.

[21] F. Pedersoli, G. Tzanetakis, and K. M. Yi, "Improving music transcription by pre-stacking A U-Net", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, 506–10.

[22] I. Pereira, F. Araújo, F. Korzeniowski, and R. Vogl, "MoisesDB: A dataset for source separation beyond 4-stems", *arxiv:2307.15913*, 2023.

[23] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "MUSDB18-HQ - an uncompressed version of MUSDB18", 2019.

[24] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks", in *Proceedings of International Conference on Machine Learning*, Vol. 97, 2019, 5301–10.

[25] S. Rouard, F. Massa, and A. Défossez, "Hybrid transformers for music source separation", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023, 1–5.

[26] K. Saito, T. Nakamura, K. Yatabe, and H. Saruwatari, "Sampling-frequency-independent convolutional layer and its application to audio source separation", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30, 2022, 2928–43.

[27] D. Samuel, A. Ganeshan, and J. Naradowsky, "Meta-learning extractors for music source separation", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, 816–20.

[28] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5), 2016, 927–39.

[29] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator", *Applied and Computational Harmonic Analysis*, 43(2), 2017, 233–68.

[30] D. Stoller, S. Ewert, and S. Dixon, "Wave-U-Net: A multi-scale neural network for end-to-end audio source separation", in *Proceedings of International Society for Music Information Retrieval Conference*, 2018, 334–40.

[31] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - a reference implementation for music source separation", *Journal of Open Source Software*, 4(41), 2019, 1667.

[32] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign", in *Proceedings of International Conference on Latent Variable Analysis and Signal Separation*, 2018, 293–305.

[33] N. Takahashi and Y. Mitsufuji, "Densely connected multi-dilated convolutional networks for dense prediction tasks", in *Proceedings of IEEE/CVF Computer Vision and Pattern Recognition Conference*, 2021, 993–1002.

[34] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band DenseNets for audio source separation", in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2017, 21–5.

[35] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains", in *Proceedings of Advances Neural Information Processing Systems*, Vol. 33, 2020, 7537–47.

[36] E. Tzinis, Z. Wang, X. Jiang, and P. Smaragdis, "Compute and memory efficient universal sound source separation", *Journal of Signal Processing Systems*, 94(2), 2022, 245–59.

[37] C. Veaux, J. Yamagishi, and K. MacDonald, "CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit", 2017.

[38] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, 2021, 2840–9.

[39] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton, "Lookahead optimizer: k steps forward, 1 step back", in *Proceedings of Advances Neural Information Processing Systems*, Vol. 32, 2019, 9597–608.