Algorithms of Sampling-Frequency-Independent Layers for Non-integer Strides

Kanami Imamura†, Tomohiko Nakamura†, Norihiro Takamune†, Kohei Yatabe‡, and Hiroshi Saruwatari†
†Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan
‡Department of Electrical Engineering and Computer Science,
Tokyo University of Agriculture and Technology, Tokyo, Japan

***Abstract*—In this paper, we propose algorithms for handling non-integer strides in sampling-frequency-independent (SFI) convolutional and transposed convolutional layers. The SFI layers have been developed for handling various sampling frequencies (SFs) by a single neural network. They are replaceable with their non-SFI counterparts and can be introduced into various network architectures. However, they could not handle some specific configurations when combined with non-SFI layers. For example, an SFI extension of Conv-TasNet, a standard audio source separation model, cannot handle some pairs of trained and target SFs because the strides of the SFI layers become non-integers. This problem cannot be solved by simple rounding or signal resampling, resulting in the significant performance degradation. To overcome this problem, we propose algorithms for handling non-integer strides by using windowed sinc interpolation. The proposed algorithms realize the continuous-time representations of features using the interpolation and enable us to sample instants with the desired stride. Experimental results on music source separation showed that the proposed algorithms outperformed the rounding- and signal-resampling-based methods at SFs lower than the trained SF.**

***Index Terms*—Sampling-frequency-independent convolutional layer, sinc interpolation, audio source separation**

I. Introduction

Deep neural networks (DNNs) have been used for various audio signal processing tasks such as music source separation [1], speech enhancement [2], and automatic music transcription [3], [4]. Most studies on DNN-based audio signal processing methods assume that the sampling frequency (SF) of an input signal is the same in the training and inference stages [1]–[16]. Hence, to handle untrained SFs, we need an additional processing such as signal resampling.

As an alternative to signal resampling, we previously proposed sampling-frequency-independent (SFI) convolutional layers [17]. These layers are based on the analogy between an ordinary convolutional layer and a collection of digital filters. A digital filter can be designed from an analog filter. Since an analog filter is SFI, we can use a collection of analog filters (latent analog filters) as an SFI structure. By utilizing a digital filter design technique, we can consistently generate the parameters (weights) of the ordinary convolutional layer from the latent analog filters for various SFs. We experimentally found that the SFI layers work more consistently than signal resampling for SFs much lower than a trained SF. Since SF is usually task-specific, the SFI layers paved the way to realize an audio source separation method that can be universally used for any downstream tasks.

The SFI layers were applied to Conv-TasNet [5], which combines DNN-based mask predictors with a trainable analysis/synthesis filterbank. This SFI extension is called SFI Conv-TasNet. It uses the SFI layers only for the analysis/synthesis filterbank, leaving the mask predictors unchanged from Conv-TasNet, which enables us to introduce recently developed mask predictors [12]–[16]. It requires that for each target SF, the kernel size K and stride S of the SFI layer should be adjusted to match those used during training in seconds. This is because the mask predictor is not SFI and its input should have the same time resolution for any SFs.

However, for some pairs of target and trained SFs, K and S become non-integers, even though a convolutional layer assumes them to be integers. For example, when SFI Conv-TasNet is trained with $K = 160$ and $S = 80$ (5 ms and 2.5 ms, respectively) at an SF of 32 kHz, the corresponding K and S are 110.25 and 55.125 at an SF of 22.05 kHz, respectively. Although the evaluation was performed at SFs where K and S became integers in [17], this problem is unavoidable to realize the versatile preprocessor of our interest.

One simple method to handle such a non-integer is to round it to the nearest integer. However, rounding S changes the time resolution of the mask predictor's input and degrades the separation performance, as we will show later in Section IV. Another method to handle non-integer strides is to use the rounding jointly with a mask predictor based on a two-dimensional convolutional neural network (2D-CNN). Paulus *et al.* experimentally showed that rounding K and S did not greatly affect the performance of speech dialogue separation when using a 2D-CNN-based mask predictor [18]. However, this method greatly restricts the architecture of the mask predictor. Specifically, the mask predictor must be agnostic to time and frequency axes and must not contain any pooling or unpooling layers. To ensure flexibility in the network architecture, we need to explore another approach.

In this paper, we propose an algorithm of the SFI convolutional layer for non-integer S by utilizing windowed sinc interpolation. This interpolation provides the continuous-time

This work was supported by JST, ACT-X Grant Number JPMJAX210G, Japan. T. Nakamura is currently with the National Institute of Advanced Industrial Science and Technology (AIST).

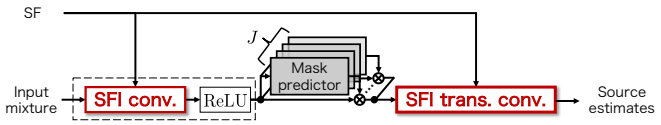


Fig. 1: Architecture of SFI Conv-TasNet. "SFI conv." and "SFI trans. conv." denote SFI convolutional and transposed convolutional layers, respectively.

counterpart of a discrete-time signal. Thus, by inserting it before the decimation in the convolutional layer, we can obtain the mask predictor's input having the desired stride. We also extend this algorithm for an SFI version of a transposed convolutional layer (SFI transposed convolutional layer). We demonstrate the effectiveness of the proposed algorithms in comparison to rounding- and signal-resampling-based methods through music source separation experiments.

II. Conventional Methods

A. SFI Convolutional Layer

We briefly describe a one-dimensional SFI convolutional layer [17]. Let $C^{(\text{in})}$ and $C^{(\text{out})}$ be the numbers of input and output channels of this layer, respectively. This layer consists of $C^{(\text{in})}C^{(\text{out})}$ analog filters and an ordinary convolutional layer. Given a target SF, it generates discrete-time impulse responses of length K that are designed to approximate the continuous-time impulse response or frequency response of the analog filters by using time- or frequency-domain filter design methods, respectively. Since the focus of this paper is how to handle non-integer S, we omitted the computation of the filter design methods due to space limitations (see [17] for details). A convolutional layer computes a cross-correlation between an input feature and the weights. Hence, the discrete-time impulse responses are reversed in time and are used as the weights for the convolutional layer. Replacing the convolutional layer with the transposed convolutional layer yields the SFI transposed convolutional layer.

B. SFI Conv-TasNet

SFI Conv-TasNet is a Conv-TasNet extension that combines J mask predictors and an *SFI* analysis/synthesis filterbank [17], where J is the number of sources. Figure 1 shows the architecture of SFI Conv-TasNet. The analysis filterbank, called the encoder, consists of an SFI convolutional layer with $C^{(\text{in})} = 1$ and $C^{(\text{out})} = C$ followed by a rectified linear unit (ReLU). Given an N-length monaural mixture $\{x[n]\}_{n=0}^{N-1}$, the encoder outputs a pseudo time-frequency representation $\{X_c[m]\}_{c=0,m=0}^{C-1,M-1}$, where M is the number of frames and n, m, and c are the discrete-time, frame, and channel indices, respectively. The pseudo time-frequency representation is fed into the mask predictor of source j, which mainly consist of one-dimensional dilated convolutional layers. See [14] for the details of the architecture of the mask predictors. After multiplying $\{X_c[m]\}_{c,m}$ by the predicted mask, we obtain a time-domain separated signal of each source $\{\hat{s}_j[n]\}_{n=0}^{N-1}$ by using the synthesis filterbank, called the decoder. The decoder is an SFI transposed convolutional layer with C input channels and one output channel. The kernel size K and stride S of the decoder are the same as those of the encoder.

At the inference stage, we need to adjust K and S in accordance with a target SF, as described in Section I. Let T be the sampling period of an input signal. To clarify the values used in the training and inference stages, we hereafter use K, S, and T for the training stage and K', S', and T' for the inference stage, respectively. The kernel size and stride at T' are given as

$$K' = \frac{T}{T'}K, \quad S' = \frac{T}{T'}S. \tag{1}$$

This adjustment keeps the time resolution of $\{X_c[m]\}_{c,m}$ unchanged, although it is valid only for integers K' and S'.

C. Windowed Sinc Interpolation

The windowed sinc interpolation is a popular bandlimited interpolation of a discrete-time signal. Let t be continuous-time and $g(t)$ be the real-valued window function. For a given discrete-time signal $x[n]$ with a sampling period of T, the interpolated signal $\tilde{x}(t)$ is given as

$$\tilde{x}(t) = \sum_{n=-\infty}^{\infty} x[n]h(t - nT, T), \tag{2}$$

where $h(t, T)$ is the windowed sinc function:

$$h(t, T) = g(t)\text{sinc}\left(\frac{t}{T}\right), \quad \text{sinc}(t) = \frac{\sin(\pi t)}{\pi t}. \tag{3}$$

Now, we choose a window function with a finite support around $t = 0$. More concretely, we assume that $g(t) = 0$ for $t < -LT/2$ or $t > LT/2$, where L is a positive integer. This choice can reduce the infinite sum in (2) to the finite sum from $n = \lceil t/T - L/2 \rceil$ to $n = \lfloor t/T + L/2 \rfloor$, which enables us to implement the interpolation with computers.

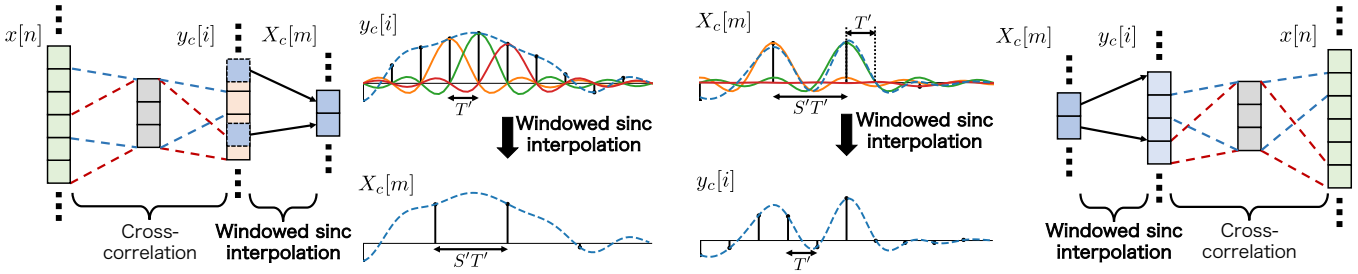
III. Proposed Method

A. Motivation and Strategy

To adjust K' and S' for a target SF, we previously used (1), as described in Section II-B. However, this adjustment method requires K' and S' to be integers. Thus, we need another method for non-integers K' and S'.

The simplest way is to round non-integers to the nearest integers. In a preliminary experiment, we confirmed that rounding K' did not significantly affect the separation performance. One reason for this is that, similar to a time-frequency transform, K' mainly affects spectral leakage in the frequency axis and has little impact on the time resolution of $\{X_c[m]\}_{c,m}$. However, S' is not used in the weight generation process and rounding S' inevitably changes the time resolution of $\{X_c[m]\}_{c,m}$. Thus, this mismatch can affect the separation performance, as we will show in Section IV.

To resolve this mismatch, we propose algorithms of the SFI layers for non-integer S' by introducing the windowed sinc interpolation. Fig. 2 shows a schematic illustration of the proposed algorithms. Since the interpolation provides a



(a) Algorithm for SFI convolutional layer

(b) Algorithm for SFI transposed convolutional layer

Fig. 2: Schematic illustration of proposed algorithms for SFI layers.

continuous-time counterpart of the discrete-time signal, it enables us to sample instants at any SF. For simplicity, we describe algorithms for the SFI layers of SFI Conv-TasNet in the subsequent sections, but the following derivations can be extended for every pair of $C^{(\text{in})}$ and $C^{(\text{out})}$. We also assume that $g(t)$ is an even function and K' is rounded, which we denote by $\check{K}'$.

B. Algorithm for SFI Convolutional Layer

In this section, we propose a sinc-interpolation-based algorithm for the SFI convolutional layer. The convolutional layer with a stride of S' and a padding size of P first computes a cross-correlation between $\{x[n]\}_n$ and the weights $\{w_c[k]\}_{c=0,k=\lfloor -(\check{K}'-1)/2\rfloor}^{C-1,\lfloor(\check{K}'-1)/2\rfloor}$ and then decimates it with an interval of S'. The cross-correlation $y_c[i]$ is given by

$$y_c[i] = \sum_{n=\lfloor -(\check{K}'-1)/2\rfloor}^{\lfloor(\check{K}'-1)/2\rfloor} x[i+n]w_c[n], \tag{4}$$

where i is the discrete-time index. The length of $y_c[i]$ is given as $I = N + 2P - \check{K}' + 1$. Although we can decimate $y_c[i]$ with a factor of S' for an integer S', we cannot perform such decimation directly for a non-integer S'. Thus, the proposed algorithm applies the sinc interpolation to $y_c[i]$, which enables us to compute $X_c[m]$ for a non-integer S'.

By invoking (2), we can describe the interpolation of $y_c[i]$:

$$\tilde{y}_c(t) = \sum_{i=\lceil t/T' - L/2\rceil}^{\lfloor t/T' + L/2\rfloor} y_c[i]h(t - iT', T'). \tag{5}$$

By sampling instants from $\tilde{y}_c(t)$ with an interval of $S'T'$, we can obtain the pseudo time-frequency representation for non-integer S'[1]:

$$X_c[m] = \tilde{y}_c(mS'T'). \tag{6}$$

If S' is an integer, (6) reduces to decimating $y_c[i]$ by a factor of S' because $\text{sinc}(t) = 0$ for any nonzero integer t.

[1]When resampling a signal with a windowed sinc interpolation, we usually avoid aliasing by changing the second argument of $h(t, T)$. However, the decimation of a convolutional layer is not followed by any antialiasing method. Thus, we emulate this decimation in (6).

C. Algorithm for SFI Transposed Convolutional Layer

Similarly to the SFI convolutional layer, we can derive a sinc-interpolation-based algorithm for the SFI transposed convolutional layer with non-integer S'. To simplify notations, we reuse $x[n]$, $y_c[i]$, and $w_c[k]$ for the counterparts of the transposed convolutional layer.

For an integer S', a transposed convolutional layer first pads $S' - 1$ zeros between $X_c[m]$ to obtain $y_c[i]$. It then outputs $x[n]$ as the cross-correlation between $y_c[i]$ and $w_c[i]$.

Now, we consider a continuous-time version of the transposed convolutional layer to handle a non-integer S'. The windowed sinc interpolation of $w_c[i]$ is given as

$$\tilde{w}_c(t) = \sum_{i=\lfloor -(\check{K}'-1)/2\rfloor}^{\lfloor(\check{K}'-1)/2\rfloor} w_c[i]h(t - iT', T'). \tag{7}$$

Since $X_c[m]$ can be seen as a discrete-time signal with a sampling period of $S'T'$, i.e., $\sum_{m=0}^{M-1} X_c[m]\delta(t - mS'T')$, we can write the continuous-time version of $x[n]$ as

$$\tilde{x}(t) = \int_{-\infty}^{\infty} \mathrm{d}\tau \sum_{m=0}^{M-1} X_c[m]\delta(t + \tau - mS'T')\tilde{w}_c(\tau) \tag{8}$$

$$= \sum_{i=\lfloor -(\check{K}'-1)/2\rfloor}^{\lfloor(\check{K}'-1)/2\rfloor} \tilde{X}_c(t + iT')w_c[i], \tag{9}$$

where $\delta(t)$ is Dirac's delta function and

$$\tilde{X}_c(t) = \sum_{m=\lceil t/(S'T') - L/(2S')\rceil}^{\lfloor t/(S'T') + L/(2S')\rfloor} X_c[m]h(t - mS'T', T'). \tag{10}$$

By sampling $\tilde{x}(t)$ with an interval of T', we obtain $x[n]$:

$$y_c[i] = \tilde{X}_c(iT'), \tag{11}$$

$$x[n] = \sum_{i=\lfloor -(\check{K}'-1)/2\rfloor}^{\lfloor(\check{K}'-1)/2\rfloor} y_c[i+n]w_c[i]. \tag{12}$$

If S' is an integer, (11) reduces to the zero-padding of $X_c[m]$ because $\text{sinc}(t) = 0$ for any nonzero integer t.

The proposed algorithms use the windowed sinc interpolation and the interpolation accuracy is determined by L. Since the interpolation accuracy would affect the separation performance, we will examine their relationship in Section IV-B.

IV. Experiments

A. Experimental Setup

To evaluate the effectiveness of the proposed algorithms, we conducted music source separation experiments using the MUSDB18-HQ dataset [19]. This dataset has 150 tracks, each of which consists of four instruments (vocals, bass, drums, and other). We used the official data split of 86, 14, and 50 tracks for training, validation, and test, respectively. As an evaluation measure, we used the signal-to-distortion ratio (SDR) obtained with the BSSEval v4 toolkit [20]. To reduce the dependency on initialization, we trained models using four random seeds and computed the averages and standard errors of the SDRs.

We used the same training setup as in [17]. As a source separation model, we chose SFI Conv-TasNet using the frequency-domain filter design, which achieved the highest separation performance in [17]. The latent analog filter was a modulated Gaussian filter. Its impulse response is given by

$$f(t) = 2\sqrt{2\sigma^2\pi}\exp\left(-\frac{\sigma^2 t^2}{2}\right)\cos(\mu t + \phi), \qquad (13)$$

where μ is the center angular frequency, σ is the parameter corresponding to the bandwidth of the filter, and ϕ is the initial phase. These parameters were initialized as in [17] and were trained jointly with other DNN parameters. The SFI Conv-TasNet was trained with a batch size of 12 for 250 epochs by RAdam [21] wrapped in a LookAhead optimizer [22]. The same data augmentations used in [17] were applied. The loss function was the negative scale-invariant source-to-noise ratio (SI-SNR) between the estimated and groundtruth signals. Since the SDR is scale-dependent, we used the scaling method presented in [14]: $\{\alpha_i\}_{j=1}^{J} = \mathrm{argmin}_{\{\alpha_j\}_{j=1}^{J}} \sum_{n=0}^{N-1}(x[n] - \sum_{j=1}^{J}\alpha_j\hat{s}_j[n])^2$, where α_j is the scale for source j.

We compared the following four methods of handling non-integer strides. *Rounding* is a method to round S' to the nearest integer. *Resampling-near* avoids non-integer S' by resampling a mixture signal at the nearest SF where the corresponding stride becomes an integer, separating the resampled signal with the trained model, and resampling the separated signals back to the original SF. *Resampling-trained* resamples a mixture signal at the trained SF. The signal resampling was implemented with `librosa` [23] as in [17]. *Proposed* uses the proposed algorithms for handling non-integer S'. The windowed sinc interpolation was implemented with `torchaudio` [24]. For $g(t)$, we used the Kaiser window with the default parameters of `torchaudio.functional.resample`. All methods used the same trained models. Following the experimental conditions used in [17], the SFs of the training and validation data were set to 32 kHz and the kernel size and stride were set to 5 and 2.5 ms ($K = 160$ and $S = 80$), respectively. The methods were evaluated at an SF of 11.025, 22.05, and 44.1 kHz, where S' becomes 27.5625, 55.125, and 110.25, respectively. In addition to these popular SFs, we evaluated the methods with an in-between SF, 16.538 kHz, to show the generality of the proposed method.

TABLE I: SDRs [dB] of Proposed with varying L for vocals

L	SF [kHz]			
	11.025	16.538	22.05	44.1
2	2.3 ± 0.3	1.5 ± 0.3	2.4 ± 0.1	1.8 ± 0.1
4	4.7 ± 0.2	4.8 ± 0.0	5.4 ± 0.1	5.5 ± 0.1
8	4.6 ± 0.2	5.2 ± 0.1	5.6 ± 0.0	5.8 ± 0.1
16	4.7 ± 0.2	5.3 ± 0.1	5.6 ± 0.1	5.8 ± 0.1
32	4.7 ± 0.2	5.4 ± 0.1	5.6 ± 0.1	5.8 ± 0.1
64	4.7 ± 0.2	5.4 ± 0.1	5.6 ± 0.1	5.8 ± 0.1

B. Relationship Between Interpolation Accuracy and Separation Performance

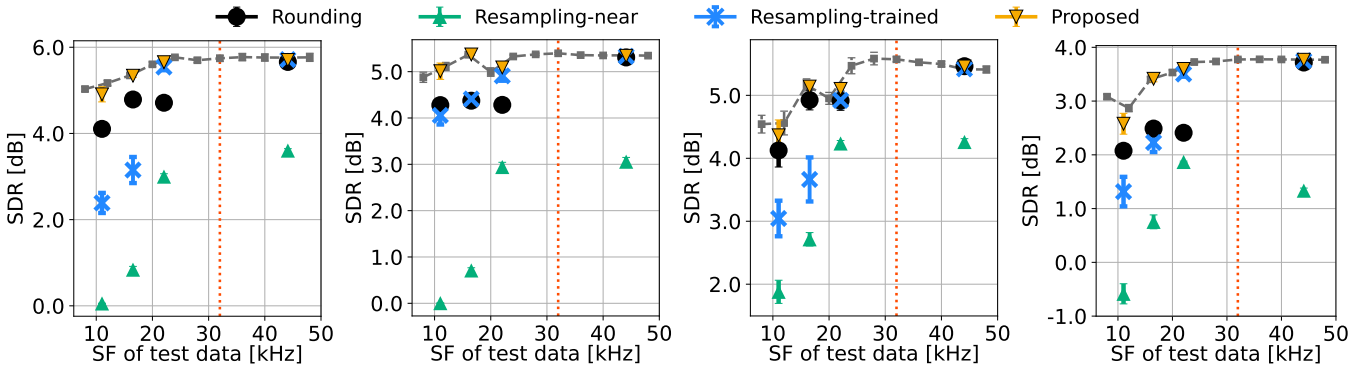
We first examined the relationship in Proposed between separation performance and interpolation accuracy. Table I lists the averages and standard errors of SDRs with varying L for vocals. The SDRs approximately increased as L increased, showing that lowering the interpolation accuracy degrades the separation performance. At all SFs, the SDRs were saturated for $L \geq 16$ and this tendency was observed in the results of bass, drums, and other. Hence, we decided to use $L = 16$.

Since L also involves the processing time, we measured the processing times for a 60-s signal using an NVIDIA GeForce RTX 3090 GPU. At an SF of 11.025 kHz, the processing time averaged over ten trials was 72.8 ms. Although this is longer than Rounding (52.1 ms), it is fast enough in practice and can be reduced by brushing up the implementation of Proposed.

C. Comparison with Existing Methods

Fig. 3 shows the averages and standard errors of SDRs for all methods per instrument. As a reference, we also show SDRs at SFs where K' and S' are integers (gray dashed curve), which we call *Reference*. SDRs of Resampling-near significantly decreased compared with Reference. Although the other methods had SDRs consistent with Reference at an SF of 44.1 kHz, Rounding and Resampling-trained quickly decreased as the SF decreased. At the lower SFs, SDRs of Rounding were around one decibel below Reference for vocals, bass, and other. One reason for the performance drop is that the change in time resolution caused by Rounding increases as the SF decreases. This result shows that rounding S' degrades the separation performance at lower SFs. Resampling-trained had higher SDRs than Rounding at an SF of 22.05 kHz. However, its SDRs dropped significantly as the SF decreases, which is consistent with the results observed in [17]. By contrast, Proposed provided more consistent SDRs with Reference at all SFs for all instruments. These results clearly demonstrate the effectiveness of the proposed algorithms.

The SDR gaps between Proposed and Rounding for vocals, bass, and other were greater than those for drums. The change in time resolution caused by rounding S' corresponds to that in the SF of an input signal for the mask predictors. Thus, it should strongly affect the separation performance for the audio signals containing pitched sounds. The drums are unpitched instruments and the change in time resolution can be less significant compared with pitched sounds. This observation



(a) vocals (b) bass (c) drums (d) other

Fig. 3: SDRs of Rounding, Resampling-near, Resampling-trained, and Proposed. Error bars represent standard errors and red dotted line denotes the trained SF. For reference, gray dashed line shows SDRs at SFs where K' and S' are integers.

leads to a conjecture that slightly changing the time resolution of an input is an adversarial attack for the separation of pitched sounds. Its investigation is beyond the scope of this paper and we leave it as future work.

V. Conclusion

We proposed algorithms of the SFI layers for non-integer strides. The proposed algorithms use the windowed sinc interpolation to bridge two signals of different SFs. For the SFI convolutional layer, the interpolation is applied to the cross-correlation between an input and the weights, which enables us to sample instants with a desired SF. For the SFI transposed convolutional layer, the interpolation is applied to an input. By using these algorithms, we can handle non-integer strides in SFI Conv-TasNet. Experimental results showed that the proposed method outperforms the rounding- and signal-resampling-based methods at lower SFs.

References

[1] 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*, vol. 1, Article 808395, p. 14, 2022.

[2] H. Dubey, V. Gopal, R. Cutler, A. Aazami, S. Matusevych, S. Braun, S. E. Eskimez, M. Thakker, T. Yoshioka, H. Gamper, and R. Aichner, "ICASSP 2022 deep noise suppression challenge," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 9271–9275.

[3] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 24, no. 5, pp. 927–939, 2016.

[4] F. Pedersoli, G. Tzanetakis, and K. M. Yi, "Improving music transcription by pre-stacking a U-Net," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 506–510.

[5] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 27, no. 8, pp. 1256–1266, 2019.

[6] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - a reference implementation for music source separation," *J. Open Source Softw.*, 2019.

[7] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *J. Open Source Softw.*, vol. 5, no. 50, p. 2154, 2020.

[8] N. Takahashi and Y. Mitsufuji, "Densely connected multi-dilated convolutional networks for dense prediction tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 993–1002.

[9] T. Nakamura, S. Kozuka, and H. Saruwatari, "Time-domain audio source separation with neural networks based on multiresolution analysis," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 29, pp. 1687–1701, 2021.

[10] A. Défossez, "Hybrid spectrogram and waveform source separation," in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2021.

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

[12] I. Kavalerov, S. Wisdom, H. Erdogan, B. Patton, K. Wilson, J. Le Roux, and J. R. Hershey, "Universal sound separation," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2019, pp. 175–179.

[13] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-Path RNN: Efficient long sequence modeling for time-domain single-channel speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 46–50.

[14] D. Samuel, A. Ganeshan, and J. Naradowsky, "Meta-learning extractors for music source separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 816–820.

[15] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 29, pp. 2840–2849, 2021.

[16] Y. Koizumi, S. Karita, S. Wisdom, H. Erdogan, J. R. Hershey, L. Jones, and M. Bacchiani, "DF-Conformer: Integrated architecture of Conv-TasNet and conformer using linear complexity self-attention for speech enhancement," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2021, pp. 161–165.

[17] K. Saito, T. Nakamura, K. Yatabe, and H. Saruwatari, "Sampling-frequency-independent convolutional layer and its application to audio source separation," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 30, pp. 2928–2943, 2022.

[18] J. Paulus and M. Torcoli, "Sampling frequency independent dialogue separation," in *Proc. Eur. Signal Process. Conf.*, 2022, pp. 160–164.

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

[20] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *Proc. Int. Conf. Latent Variable Anal. Signal Separation*, 2018, pp. 293–305.

[21] 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 *Proc. Int. Conf. Learn. Representations*, 2020.

[22] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton, "Lookahead optimizer: k steps forward, 1 step back," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 9597–9608.

[23] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. Python Sci. Conf.*, vol. 8, 2015, pp. 18–25.

[24] Y.-Y. Yang, M. Hira, Z. Ni, A. Astafurov, C. Chen, C. Puhrsch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang, J. Lian, J. Hwang, J. Chen, P. Goldsborough, S. Narenthiran, S. Watanabe, S. Chintala, and V. Quenneville-Bélair, "TorchAudio: Building blocks for audio and speech processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 6982–6986.