MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Embracing Cacophony: Explaining and Improving Random Mixing in Music Source Separation

Jeon, Chang-Bin; Wichern, Gordon; Germain, François G; Le Roux, Jonathan

TR2026-012 January 14, 2026

Abstract

In music source separation, a standard data augmentation technique involves creating new training examples by randomly combining instrument stems from different songs. However, these randomly mixed samples lack the natural coherence of real music, as their stems do not share a consistent beat or tonality, often resulting in a cacophony. Despite this apparent distribution shift, random mixing has been widely adopted due to its effectiveness. In this work, we investigate why random mixing improves performance when training a state-of-the-art music source separation model and analyze the factors that cause performance gains to plateau despite the theoretically limitless number of possible combinations. We further explore the impact of beat and tonality mismatches on separation performance. Beyond analyzing random mixing, we introduce ways to further enhance its effectiveness. First, we explore a multi-segment sampling strategy that increases the diversity of training examples by selecting multiple segments for the target source. Second, we incorporate a digital parametric equalizer, a fundamental tool in music production, to maximize the timbral diversity of random mixes. Our experiments demonstrate that a model trained with only 100 songs from the MUSDB18-HQ dataset, combined with our proposed methods, achieves competitive performance to a BS-RNN model trained with 1,750 additional songs

IEEE Open Journal of Signal Processing 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/XXXX.2022.1234567

Embracing Cacophony: Explaining and Improving Random Mixing in Music Source Separation

Chang-Bin Jeon[1], Member, IEEE, Gordon Wichern[1], Member, IEEE, François G. Germain[1], Member, IEEE, and Jonathan Le Roux[1], Fellow, IEEE

[1]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139 USA

Corresponding author: Gordon Wichern (email: wichern@merl.com).

This work was performed while C.-B. Jeon was an intern at MERL. He is currently affiliated with Samsung Electronics.

ABSTRACT In music source separation, a standard data augmentation technique involves creating new training examples by randomly combining instrument stems from different songs. However, these randomly mixed samples lack the natural coherence of real music, as their stems do not share a consistent beat or tonality, often resulting in a cacophony. Despite this apparent distribution shift, random mixing has been widely adopted due to its effectiveness. In this work, we investigate why random mixing improves performance when training a state-of-the-art music source separation model and analyze the factors that cause performance gains to plateau despite the theoretically limitless number of possible combinations. We further explore the impact of beat and tonality mismatches on separation performance. Beyond analyzing random mixing, we introduce ways to further enhance its effectiveness. First, we explore a multi-segment sampling strategy that increases the diversity of training examples by selecting multiple segments for the target source. Second, we incorporate a digital parametric equalizer, a fundamental tool in music production, to maximize the timbral diversity of random mixes. Our experiments demonstrate that a model trained with only 100 songs from the MUSDB18-HQ dataset, combined with our proposed methods, achieves competitive performance to a BS-RNN model trained with 1,750 additional songs.

INDEX TERMS Music source separation, random mixing, data augmentation

I. INTRODUCTION

MUSIC source separation (MSS) has seen tremendous progress in performance during the deep learning era, with an ever-growing list of high-performing models in both the open-source community and commercial applications [1], [2]. While this success is often credited to advances in network architectures [3]–[5] and the increasing availability of public datasets [6]–[8], the contribution of training pipeline improvements (e.g., data augmentation) has been largely overlooked and remains poorly understood. Meanwhile, MSS research continues to suffer from a chronic lack of large-scale training data. Despite recent efforts to leverage self-supervised training [9], state-of-the-art MSS models still require access to substantial supervised data, where stems (i.e., recordings of the isolated sources) are available as training targets. This remains challenging due to practical barriers in recording such data (especially outside professional studios) and intellectual property concerns. As a result, MSS datasets are typically small, with MUSDB18-HQ [6], the most widely used dataset in the field, containing only 150 songs—approximately 10 hours per stem.

Data augmentation techniques have long been employed to improve performance, particularly under conditions of limited training data. Among these, methods relying on the random combination of samples have become popular in various domains due to their simplicity and effectiveness. For example, approaches like mixup [10] or cutmix [11] have been successfully applied to train classifier systems, including audio models [12]. Similarly, random mixing—combining randomly selected audio sources—is a staple in learning-based audio source separation, spanning applications such as speech enhancement and separation [13]–[15], natural sound separation [16], and soundtrack separation [17]. Importantly, in most of these applications, random mixes result in plausi-

ble sound scenes. The fact that these mixes are conceptually ”in-distribution” with respect to the target task intuitively justifies their effectiveness during training.

In the MSS literature, however, random mixing often creates “song snippets” by combining stems from different songs to form additional training examples [18]. While widely effective, this technique introduces an apparent paradox: during inference, an MSS system will typically be asked to separate songs with stems that are consistent in terms of musical attributes such as beat or key. Randomly mixed stems lack such consistency, often resulting in cacophonous, “out-of-domain” mixes relative to the target task. Nevertheless, random mixing remains standard practice, particularly in the widely adopted 4-stem separation tasks (vocals, bass, drums, and other) used in recent challenges [19]–[21]. This raises questions about the underlying mechanisms driving its success and its limitations. Furthermore, recent work on multi-singer separation [22] highlights the unclear justification for using random mixing in scenarios where source consistency (e.g., timbre, pitch, and time) is crucial.

To mitigate this domain mismatch, researchers have proposed alternative or complementary data augmentation techniques. Synthesized stems generated from publicly available MIDI songs [7] produce diverse, consistent mixtures, while pitch-shifting and time-stretching combined with random mixing [4] aim to align mixtures more closely in key and beat. Despite these efforts, these methods have yet to consistently outperform random mixing alone. In our previous paper [23], we first aimed to clarify why random mixing remains effective despite the distribution shift it introduces by considering the following research questions:

- Why do networks trained with randomly mixed examples outperform those trained without random mixing, despite the apparent distribution shift?
- While random mixing can generate limitless training data, performance improvements eventually plateau. To what extent does random mixing yield diminishing returns?
- Consistent beat and tonality are key differences between random and original mixes. How do these characteristics affect MSS performance?

In this paper, we investigate each of these questions in greater depths, and extend our analysis of random mixing to further include a more practical question: *how can the effectiveness of random mixing be further improved?*

To address these questions, we analyze the training dynamics of MSS networks with and without random mixing. We explore the effective number of unique random mixes during training and evaluate how variations in beat and tonality across instrument stems affect performance using pitch- and time-shifted test data. Furthermore, we propose leveraging a parametric equalizer (EQ) to further increase the timbre variation of randomly mixed data. Finally, we examine a multiple-sampling strategy for random mixing that significantly diversifies source combinations.

This paper focuses on the 4-stem separation task using MUSDB18-HQ, where stems are correlated in pitch and time but differ in timbral characteristics. While we leave the analysis of other MSS tasks for future work, we believe our findings and conclusions provide valuable insights into the role of random mixing across diverse separation contexts.

Code for our data augmentation pipeline will be available online[1].

II. Music Source Separation

A. Task Definition

The task of music source separation consists in extracting the signal of individual sources (typically referred to as *stems*) from a musical input mixture. Mathematically, we consider a mixture signal x as the sum of N stems $x^{(n)}$, i.e.,

$$x = \sum_{n=1}^{N} x^{(n)}. \tag{1}$$

The task consists then in putting together a system (e.g., an MSS model) which, from input x, generates its best estimate $\hat{x}^{(n)}$ of $x^{(n)}$ for each n. We note that the definition of “source” is ambiguous, and musical songs include different numbers and types of instruments. Subsequently, we adopt the simplifying convention from the typical 4-stem MSS task [19]–[21] where we define sources as belonging to a fixed closed set of $N = 4$ distinct source types, i.e., *vocals, drums, bass*, and *other*. The signal from a given musical instrument is then found in the stem of its corresponding source type. We also adopt the typical convention that all signals are stereo (i.e., 2-channel) signals.

B. Experimental Setup: Models

TFC-TDF-UNet v3: To make our analysis as current as possible, we use the state-of-the-art TFC-TDF-UNet v3 architecture [24], winner of the recent 2023 MDX Challenge Leaderboard A [21]. The model takes as input the short-time Fourier transform (STFT) of the mixture signal x, outputs an estimated STFT for all sources, and is trained using mean-squared error loss between the waveforms of the estimated stems (obtained as the inverse STFT of the network output) and the ground-truth stem waveforms. In our experiments, we use the same TFC-TDF-UNet v3 settings and hyperparameters that were used to train the model on the MUSDB18-HQ dataset in the original paper, including the use of 6-second training chunks with completely silent target chunks removed, overlap-add at inference time, model exponential moving average (EMA) [25], and automatic mixed precision training [26]. We only change the batch size to 4 to fit the model on a single 24 GB memory GPU. Every model instance was trained for 470k steps. Additionally, we employ loudness normalization [27] to minimize the domain mismatch in terms of loudness, by normalizing all mixes to -14 LUFS without applying a

[1] https://github.com/merlresearch/embracing-cacophony

limiter. We also use pitch-shift data augmentation [28] on all training mixes to best match the state-of-the-art configuration in [24]. Specifically, we randomly select a pitch shift in $\{-3, -2, -1, 0, 1, 2, 3\}$ semitones to be applied to each stem, in an *inconsistent* manner. When this augmentation is applied to the original mixes, we adopt a *consistent* strategy where the same pitch shift amount is uniformly applied to all the stems. A comparative analysis between the consistent and inconsistent strategies is presented in Section IV.

Open-Unmix: We also train Open-Unmix [1] for a few experiments to check whether our findings on TFC-TDF-UNet v3 also apply to other architectures. Unlike TFC-TDF-UNet v3, which is a CNN-based architecture that directly uses the complex spectrogram as input and output, Open-Unmix operates on the magnitude spectrogram, produces a magnitude mask that is combined with the mixture phase to produce a source estimate, and consists of three Bi-LSTM layers with skip connections. We follow the training setting of the original implementation[2] and use the early stopping based on the validation loss; specifically, we stop the training when the validation loss does not decrease for 80 epochs, where 1 epoch consists of 64 iterations. The batch size for each iteration is 16 and we use a single 24 GB memory GPU to match the experiments for TFC-TDF-UNet v3. We also add pitch-shift data augmentation [28] to the Open-Unmix training setup to maintain parity with the TFC-TDF-UNet v3 recipe.

C. Experimental Setup: Data and Metrics

MUSDB18-HQ [6] consists of 150 songs split into a 100 song training set and a 50 song test set (3.5 hours). Following common practice, we divide the training set using 86 songs for training (5.3 hours) and 14 for validation (1.0 hours).

We use the source-to-distortion ratio (SDR) [29] implemented in museval [19] and median of frames, median of tracks, which is a standard criteria in 4-stem music source separation [19]. However, for the few experiments (Fig. 2) that require hundreds of evaluations for every 10k training steps, we use instead the more computationally efficient SDR metric proposed at the recent SDX challenge [21], which is defined as,

$$\overline{\text{SDR}}^{(n)} = 10 \log_{10} \frac{||x_L^{(n)}||^2 + ||x_R^{(n)}||^2}{||x_L^{(n)} - \hat{x}_L^{(n)}||^2 + ||x_R^{(n)} - \hat{x}_R^{(n)}||^2}, \quad (2)$$

where $x_L^{(n)}$ and $x_R^{(n)}$ stand for the target source of the left and right channel of stem n, respectively. In this paper, we denote this score as $\overline{\text{SDR}}$. Except in Fig. 1 and Fig. 4, all scores are averaged across the 4 stems.

D. Random Mixing Augmentation

Since music with isolated stem data is difficult to acquire, researchers have used various data augmentation methods to maximize the potential of small public datasets. As described in the introduction, the focus of this paper is random mixing augmentation [18]. Formally, we have a dataset $\mathcal{S} = \{s_m, m = 1, \ldots, M\}$ of M songs, where each song s_m is composed of N stems $s_m^{(n)}$, following the relationship in (1), i.e., $s_m(t) = \sum_{n=1}^{N} s_m^{(n)}(t), \forall t$. During training, we create *original mixes* by sampling, for the i-th training sample, a song m_i and a start time τ_i, defining a mixture x_i and corresponding stems $x_i^{(n)}$ as

$$x_i(t) = \sum_{n=1}^{N} x_i^{(n)}(t) = \sum_{n=1}^{N} s_{m_i}^{(n)}(t + \tau_i), \quad (3)$$

where $t \in [0, T]$ and T corresponds here to a 6 second excerpt. Note that, in (3), the start time τ_i is constant across stems and all stems come from the same song indexed by m_i.

Alternatively, we can create *random mixes* during training by sampling a different song $m_{i,n}$ and a different start time $\tau_{i,n}$ for each stem n, defining the i-th training sample as

$$x_i(t) = \sum_{n=1}^{N} x_i^{(n)}(t) = \sum_{n=1}^{N} s_{m_{i,n}}^{(n)}(t + \tau_{i,n}). \quad (4)$$

This enables the creation of unique and varied input random mixtures at each training iteration, even with a small amount of stem data.

III. Should we include original mixes at all?

A. Influence of the Ratio of Original Mixes

We experiment with various combinations of original and random mixes during model training to investigate how original mixes contribute to a network's final performance. Specifically, we set a probability p for each experiment such that each mix in a batch of training data is sampled as an original mix with probability p and as a random mix with probability $1-p$.

When random mixing was first used in the context of music source separation [18], the performance gain was found to be only $0.2\,\text{dB}$ SDR (BLSTM-1 in Table 2 of [18]). However, we observe a $2.1\,\text{dB}$ gain when training with only random mixes ($p=0.0$) compared to training with only original mixes ($p=1.0$) (see Fig. 1). We contend that this is due to a much more expressive model architecture: TFC-TDF-UNet v3 is a deep U-Net with 70M parameters, whereas BLSTM-1 is a shallow recurrent network with a single BLSTM layer. This allows our model to learn more diverse features from the infinitely many augmented mixes. Somewhat surprisingly, performance degrades consistently as more original mixes are added to the training data, starting from the the $p = 0.0$ model. While one may have thought that adding some amount of musically realistic "in-domain" data would have been beneficial, this does not appear to be the case for this model.

Interestingly, in Fig. 1, using random mixes even with only a probability of 0.1 ($p = 0.9$) results in a substantial performance gain of $1.0\,\text{dB}$. Since we train models with batch size $B=4$ for $I=470\text{k}$ iterations, a total of $1\,880\,000$

[2]https://github.com/sigsep/open-unmix-pytorch

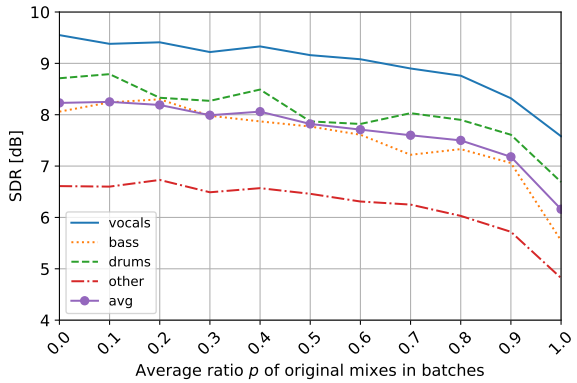
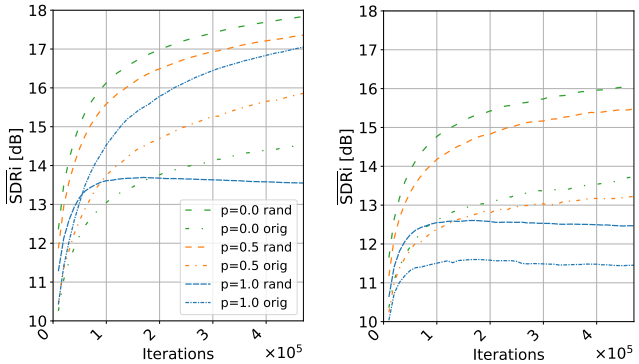


FIGURE 1. Test SDR for TFC-TDF-UNet v3 networks trained with different ratios of original and random mixes. '$p=1.0$' corresponds to the model trained solely with original mixes and '$p=0.0$' to the model trained solely with random mixes.



(a) Training set mixes (b) Validation set mixes

FIGURE 2. Average $\overline{\text{SDR}}$ improvement during training as measured on sets of random (dashed lines) and original mixes (dash-dotted lines) for three different models. In the legend, 'rand' and 'orig' mean sets of mixes that the SDRi scores were measured on for each model. For example, '$p=0.5$ orig' is a model trained using $p=0.5$, evaluated every 10k iterations on original mixes. The model trained with only original mixes ($p=1.0$) overfits to the training data. Other models trained with random mixes ($p=0.5$ and $p=0.0$) do not.

different mixes are generated over the duration of the training. That is, the $p=0.9$ model has learned significantly meaningful information from only 188 000 randomly mixed training examples. This leads to the necessity for further investigations on the effective number of random mixes, which we will discuss in Section III.C.

B. Training Dynamics Comparison

In Fig. 2, we compare the performance of three models ($p \in \{1.0, 0.5, 0.0\}$) over the course of training, measured every 10k iterations. Performance is evaluated on four sets consisting of either original mixes or random mixes, generated from either training or validation data. These sets are generated once and fixed across training iterations. To match the training setting, evaluations are performed on 6-second chunks (for original mixes, songs are split without overlap). For each stem's evaluation, random mixes are generated independently across 3 different seeds to marginalize the influence of randomness, resulting in twelve times as many chunks as the original mixes.

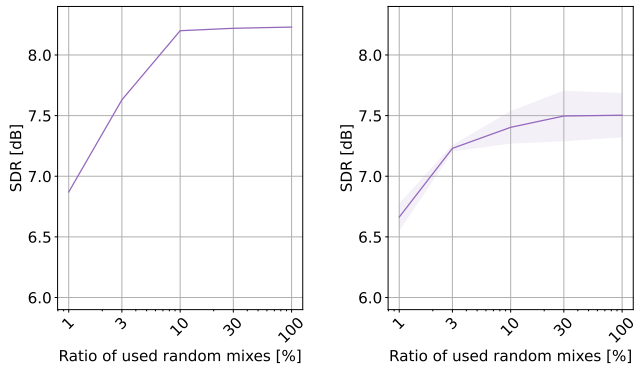
Noticeably, for the model only trained with original mixes ($p=1.0$), shown in the blue curves in Fig. 2, the scores on original mixes in the training set are increasing while the scores on random mixes in the training set and both conditions in the validation set are decreasing. That is, the model trained without using random mixes easily overfits to the training data. On the other hand, models trained with at least some random mixes ($p=0.5$ and $p=0.0$, shown in the orange curves with the medium gap and the green curves with the largest gap, respectively) do not appear to overfit.

In Fig. 2(a), we also observe that the $p=0.5$ model (the medium gap dashed orange curve) performs better than the $p=0.0$ model (the largest gap dashed green curve) on the original mixes from the training set, while performing worse on the validation original mixes in Fig. 2(b). We conjecture this is because the $p=0.5$ model has seen the same mixes too many times during training, which we explore in more detail in Section III.C.

C. Influence of the Number and Variety of Random Mixes

In this section, we investigate the amount of random mixes needed for effective training. first define a number N_R of unique random mixes as a fraction R (in %) of the total number $I \times B$ of training examples (which as discussed previously is 1 880 000). We then train models by sampling training examples only from those fixed N_R random mixes, repeating them throughout training until we reach $I \times B$ samples, without using any original mixes. In Fig. 3(a), we see that even when the model is trained with only $R=10\%$ of the total random mixes (i.e., $N_R = 188\,000$), there is almost no decrease in performance. This result somewhat explains the performance gain of the $p=0.9$ model in Fig. 1, where 188 000 random mixes used in conjunction with original mixes were actually enough to train a model with competitive performance. On the other hand, it is also noticeable that the final performance of the $p=0.9$ model from Fig. 1 (7.2 dB SDR) is far below that of the model trained solely by repeating the 10% of fixed random mixes from Fig. 3(a) (8.2 dB). This likely indicates that the model trained with $p=0.9$ original mixes overfits to the original mixes. It is also possible that original mixes are less informative for model training, which we will investigate further in Section IV.

In Fig. 3(b), we create the same fixed number of random mixes N_R, but sample training stems from a reduced dataset obtained by randomly selecting about half of the songs from the 86 training songs in MUSDB18-HQ. We repeat the song selection and random mix creation with 3 different random seeds, where the reduced dataset's length was 2.3, 2.6, and 3.0 hours. The shaded regions in the figure show the standard deviation across seeds. Similarly to Fig. 3(a), Fig. 3(b) also confirms that training with 10% of fixed random mixes is comparable to using a unique random mix for each training sample (100%). Furthermore, it is noteworthy that



(a) With full data (b) With reduced data

FIGURE 3. Test SDR when varying the amount of random mixes created from (a) the full training data and (b) a reduced set of songs. Models are trained by repeatedly sampling batches from this fixed set of random mixes for $I = 470$k iterations, with the size of this set indicated on the x-axis as a ratio R % to the total number of training examples.

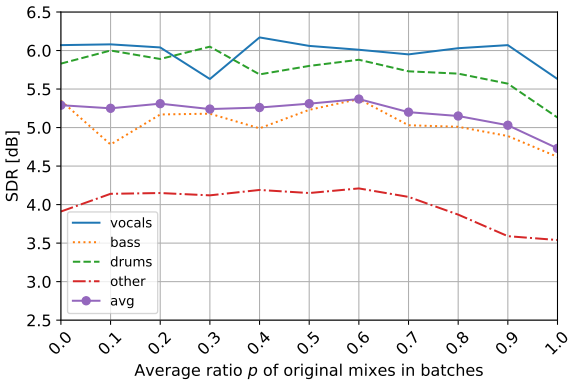


FIGURE 4. Test SDR for Open-Unmix networks trained with different ratios of original and random mixes. Compared to Fig. 1, the highest performance on *avg* is observed at $p = 0.6$ rather than $p = 0.0$ but there was no significant difference in performance in the range from $p = 0.0$ to $p = 0.6$. Additionally, the performance gap between $p = 0.0$ and $p = 1.0$ was not as large as in Fig. 1.

just repeatedly using 90 hours of random mixes obtained from 86 songs (the 3% setting in Fig. 3(a)) performs on par or better than using about 3000 hours of never-repeating random mixes from the reduced song data (100% setting in Fig. 3(b)). It thus appears that a small amount of random mixes composed of stems from a larger set of songs performs better than a large amount of random mixes composed of stems from a smaller set of songs. Thus, the diversity or novelty of random mixes may be more important than their amount; data augmentation alone is not enough, fresh data is necessary for future improvements.

D. Applicability to Other Models

To validate that the presented results extend beyond the TFC-TDF-UNet v3 architecture, we perform an additional training experiment where we vary p similar to Fig. 1, this time using the popular Open-Unmix network [1]. The behavior shown in Fig. 4 is largely similar to the result from TFC-TDF-UNet v3 (Fig 1), with the model benefiting from the inclusion of random mixes in the training data, as performance improves when starting from $p = 1.0$ and decreasing p. However, in contrast with TFC-TDF-UNet v3 where performance steadily increase and best performance was achieved at $p = 0.0$, performance for Open-Unmix saturates at $p = 0.6$ (5.4 dB SDR on *avg*) then remains roughly steady, obtaining 5.3 dB at $p = 0.0$. We speculate this difference may be due to the smaller size of Open-Unmix, which has only 14M trainable parameters, compared to 70M for TFC-TDF-UNet v3. That is, a larger model capacity may be able to better take advantage of the increased data diversity from random mixing. It is also noteworthy that the result at $p = 0.0$ from Open-Unmix (5.3 dB) does not outperform the original result (5.4 dB) reported in the paper [1], even though we train the model with additional pitch-shift augmentation [28] while keeping all other settings the same. Again, we speculate that Open-Unmix may not have the capacity to benefit from pitch-shift augmentation during training, while larger models such as TFC-TDF-UNet v3 do benefit from such augmentations.

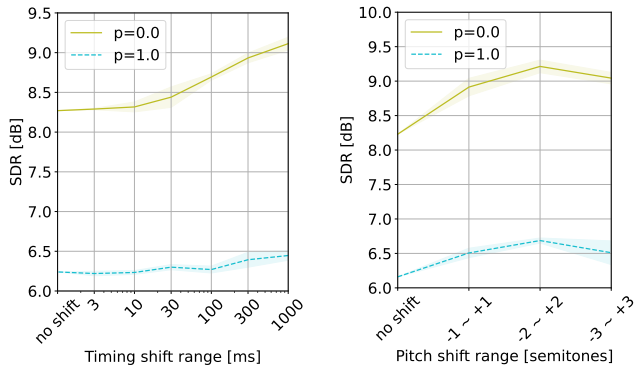
IV. Influence of Consistent Beat and Tonality

In this section, we analyze how consistent beat and tonality across different mixture stems affect performance in music source separation.

A. Influence of Consistent Beat and Tonality in Evaluation

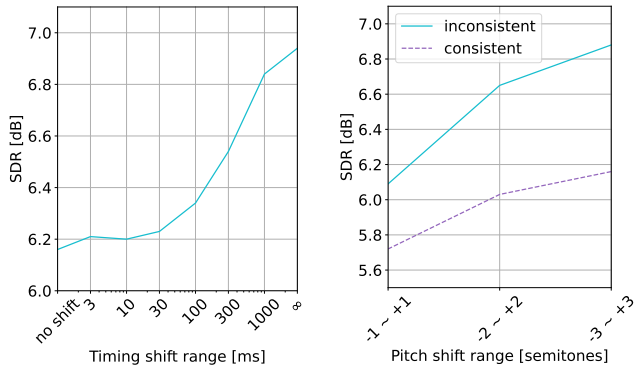
In Fig. 5, on the test set of MUSDB18-HQ, we apply (a) timing shift and (b) pitch shift to each stem to form mixtures that have (a) inconsistent beat and (b) inconsistent tonality, respectively, even as all stems are from the original song. Specifically, timing shift parameters for each stem are sampled from uniform distribution $\mathcal{U}(-t_s, t_s)$, where t_s is the range of timing shift (x-axis in Fig. 5(a)). Pitch shift values in semitones are discretely sampled up to a maximum shift of 3 semitones up or down, as shown in the x-axis in Fig. 5(b). For each parameter, 3 different random seeds are used. Then, we separate the mixtures using the $p = 0.0$ and $p = 1.0$ models from Fig. 1, i.e., models trained with only random and only original mixes, respectively.

In Fig. 5(a), we observe that test sets with inconsistent beat are easier to separate (i.e., higher SDR). Even when timing shift parameters are smaller than STFT parameters (e.g., a shift of $t_s = 30$ ms is smaller than both the 186 ms FFT size and 46 ms hop size of the model), performance starts to increase. When $t_s = 1$ s, the performance gain is almost 0.9 dB. We also confirm that inconsistent tonality from pitch shifting makes source separation much easier in Fig. 5(b), as we observe improvements up to 1.0 dB. In addition, it is noteworthy that the $p = 0.0$ model benefits more than the $p = 1.0$ model in these experiments. This should be expected, because the $p = 0.0$ model was trained on random



(a) Inconsistent beat (b) Inconsistent tonality

FIGURE 5. **Test SDR for various (a) timing and (b) pitch modified test sets using the models trained with only original mixes ($p=1.0$) or only random mixes ($p=0.0$) from Fig. 1. Solid lines show the average across test sets generated from 3 different random seeds, and the shaded regions show the standard deviation.**



(a) Inconsistent beat (b) Inconsistent tonality

FIGURE 6. **Test SDR of models trained on original mixes with (a) slightly shifted timing or (b) inconsistent tonality across stems.**

mixes with inconsistent beat and tonality while the $p = 1.0$ model was not, so the distribution shift between training and test data would be much smaller for the $p = 0.0$ model.

B. Influence of Consistent Beat and Tonality in Training

Furthermore, we investigate the effects of timing and pitch shift during training, but evaluated on the unmodified MUSDB18-HQ test set. In Fig. 6(a), we train models without random mixing but with slight timing shifts applied to each stem during training, and add the $t_s = \infty$ case, where 6-second training samples can come from anywhere in the same song, i.e., within-song random mixing. We observe that, for shifts greater than $100\,\mathrm{ms}$, which is longer than the STFT hop size, the scores of the models start to increase, reaching $6.8\,\mathrm{dB}$ SDR at $t_s = 1\,\mathrm{s}$ shift. Furthermore, while within-song random mixing ($t_s = \infty$) shows the highest $6.9\,\mathrm{dB}$ SDR, as expected given that it allows for the most diversity, it is only $0.1\,\mathrm{dB}$ higher than the $t_s = 1\,\mathrm{s}$ shift. Compared to the full random mixing across songs ($p = 0.0$ model), which obtains $8.2\,\mathrm{dB}$ SDR, it is notable that within-song random mixing leads to a significant performance decrease of $1.3\,\mathrm{dB}$, underlining the impact of increasing the diversity of the pool of songs from which random mixes are created.

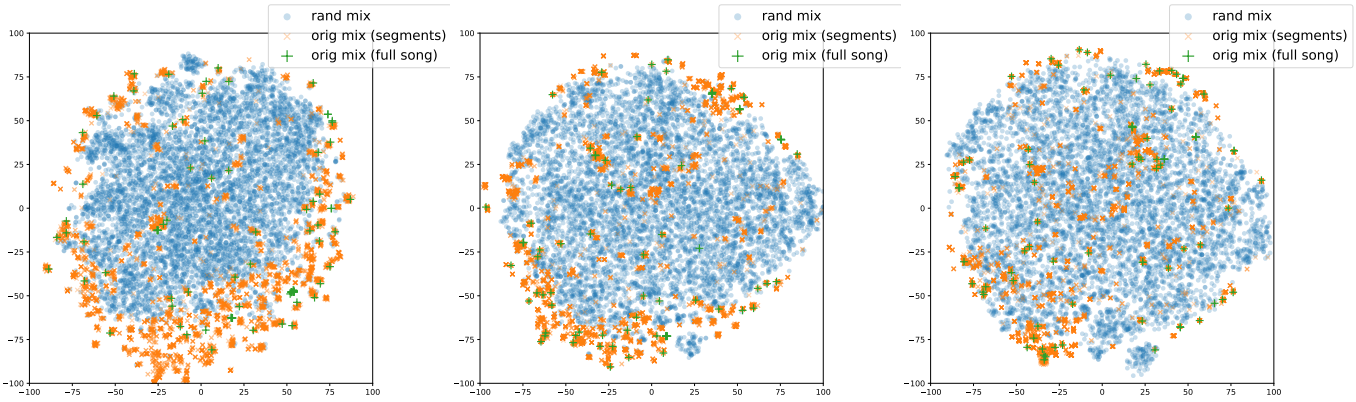
Fig. 6(b) displays results after training with only original mixes, but applying pitch shifts to each stem in both a 'consistent' manner (same pitch shift applied to each stem, see Section II.B and an 'inconsistent' manner (different pitch shift applied to each stem). Scores increase with larger amounts of pitch shift for both cases, with inconsistent pitch shift providing a notable improvement. The best consistent pitch shifting ($6.2\,\mathrm{dB}$, obtained for the largest pitch shift range, which corresponds to the $p=1.0$ model from Fig. 1) is much worse than the best inconsistent one, suggesting that inconsistent tonality between stems during training is an important component of random mixing. Still, we note that even the best-scoring configurations of within-song random mixing and original mixes with inconsistent tonality in Fig. 6(a) and Fig. 6(b) score clearly worse than full random mixing even when random mixing was used the least in our experiments ($p = 0.9$ in Fig. 1).

TABLE 1. **Test SDR (mean $\pm$ standard deviation in [dB]) for models trained with 2.7 hours of data consisting in chunks from original songs (i.e., "original mixes" with consistent beat and tonality), chunks from randomly made songs (song-level mixing of stems with inconsistent beat and tonality), and $R = 0.09\%$ of fixed random mixes (which corresponds to the same amount of data as the previous two cases but obtained from a wider variety of stems)**

Training data	*vocals*	*bass*	*drums*	*other*	*avg*
Original	7.4 ± 0.1	5.4 ± 0.1	6.6 ± 0.1	4.7 ± 0.1	6.1 ± 0.1
Random (song-level)	-0.9 ± 0.9	-1.4 ± 1.0	2.2 ± 2.2	3.0 ± 0.6	0.7 ± 1.1
Random ($R = 0.09\%$)	5.4 ± 0.1	5.1 ± 0.1	5.9 ± 0.1	4.0 ± 0.1	5.1 ± 0.0

C. When Does Domain Consistency Matter?: Training with Song-Level Random Mixes

In Table 1, to further investigate how domain mismatch affects the performance of MSS networks, we randomly choose stems from different songs in the train set of MUSDB18-HQ and mix them to make a dataset of 86 beat- and tonality-inconsistent songs from which training chunks will be sampled (until now, random mixing was at the 6-second chunk level, resulting in many more possibilities). For each song, the four chosen stems are trimmed to match the shortest one. This results in new datasets of *random songs* or *song-level random mixes* across 3 random seeds that each have length of 2.7 hours. Then, we train models on the random songs without random mixing (similarly to the $p = 1.0$ model in Fig. 1) and compare them to models trained on the original songs. For fair comparison, we use 3 random 2.7 hour subsets obtained by randomly trimming each original song. Lastly, we train models with 2.7 hours of fixed random mixes, which corresponds to the $R = 0.09$ % condition from the experiment in Fig. 3, to compare the



(a) $R = 1\%$ (b) $R = 3\%$ (c) $R = 10\%$

FIGURE 7. T-SNE plots of original and fixed random mixes with varying ratios of fixed random mixes. Blue dots represent random mixes, while orange 'x' markers and green crosses indicate original mixes at the segment level and song level, respectively. Notably, as the ratio increases, the distributions of original and fixed random mixes exhibit greater overlap. This supports the assumption that a sufficiently large number of random mixes can approximate the characteristics of original music throughout training. Furthermore, by exposing models to interpolated training examples, a sufficiently diverse set of random mixes can potentially enhance final performance.

performance of the models trained with random songs and the same amount of random mixes.

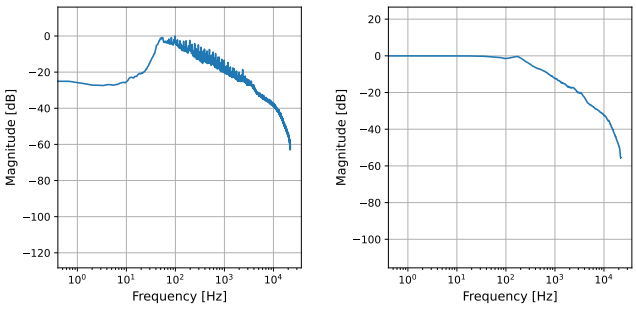
Surprisingly, data with inconsistent beat and tonality (*Random (song-level)*) leads to significant performance degradation compared to consistent beat and tonality (*Original*), dropping from $6.1\,\mathrm{dB}$ to $0.7\,\mathrm{dB}$ SDR. In contrast, a model trained using 2.7 hours of fixed random mixes obtained at the 6-second chunk level ($R = 0.09\%$, the last row in Table 1), where the training examples are likely much more diverse, obtained $5.1\,\mathrm{dB}$ SDR. This implies that the domain consistency matters in music source separation, all the more so as data diversity becomes limited. This could explain why, in the task of vocal harmony separation where consistency is arguably even stronger, random mixing was found to be detrimental on a small dataset (104 minutes) in [30], but significantly beneficial on a much larger one (400 hours) in [22].

D. Why Does Music Source Separation Benefit from Cacophony?

Until now, we have confirmed how consistent beat and tonality of a mixture affect music source separation in both the test phase (Fig. 5) and training phase (Fig. 6 and Table 1). Furthermore, we have observed that musical (i.e., non-random) mixes can be beneficial when data is limited (Table 1). However, it is still unclear why random mixing augmentation is so powerful in music source separation. To gain further insight, we conduct a visual analysis using a recent audio representation learning model.

In Fig. 7, we extract the CLAP [31] vector representations of random mixes (6-second chunks) corresponding to 1%, 3%, and 10% of the total number of training samples as well as those of the fixed original mixes (6-second chunks), and then visualize them through t-SNE [32]. We also include the full song original mix embeddings, which, following the default approach from [31], consist of the three 10-second random chunks from a full song (from the front 1/3, middle 1/3, back 1/3 position) and one chunk with the entire audio file downsampled to length equivalent to 10 seconds. The low-dimensional visualizations in each subplot are fit separately using both original and random mixes. We use CLAP embeddings because CLAP provides task-independent audio representations that capture the general acoustic and semantic characteristics of music, and provides better subjective score than competing embedding models in [33]. Thus, we believe it is suitable for analyzing distributional similarity between random and original mixes in a model-agnostic manner. We chose t-SNE because of its ability to cleanly visualize clusters compared to competing techniques such as UMAP [34], as we first expected that random and original mixes would form clearly separated clusters due to their distinctive nature in terms of beat and tonal consistency. However, as can be seen in Fig. 7(c), it is clearly noticeable that original mixes are spread over the distribution of random mixes. At the same time, we can consider that random mixes are covering the overall distribution of original mixes while also interpolating diverse small clusters of original mixes.

Through the lens of these t-SNE plots, it is also possible to explain the results that we previously discussed in Section III.C, where we saw that, when the data is limited, for example when the data pool where random mixes come from is small or the number of random mixes is small, the diversity of random mixes is reduced. Indeed, when comparing Fig. 7(a) and Fig. 7(b) to Fig. 7(c), we can clearly confirm that, as the diversity decreases by using a smaller percentage R of random mixes, the number of random mixes that interpolate the small clusters of original mixes is decreased and the random mixes form independent clusters rather than interpolating original mixes. Those independent



(a) (b)

FIGURE 8. Average frequency curves of (a) the 86 songs in the training dataset and (b) their spectra after applying a Savitsky-Golay filter. Shaded regions depict the standard deviations.

clusters are likely less helpful than original mixes for final performance due to domain mismatch.

V. Influence of Timbre

Timbre, the color of tone, is also one of the main aspects that constitutes music along with beat and tonality. Nevertheless, it is difficult to analyze the effect of timbre in music source separation because timbre is an abstract characteristic rather than a computationally defined one, and using random mixes is already introducing diverse combinations of timbre to models. In this section, we raise two specific research questions regarding timbre. *(i)* Do models learn and get familiar with the overall timbre of training data? If so, would it be helpful to explicitly manipulate the timbre of test inputs so that it matches the average timbre of training data? *(ii)* Does explicitly manipulating timbre during training benefit music source separation performance? To this end, we experiment with directly manipulating the frequency response of network inputs using a parametric equalizer (EQ) during inference (Section V.A) and training (Section V.B). Note that what we mainly manipulate here is the average frequency response of a song, a feature that is highly correlated with timbre but may be slightly different because it does not contain time-varying characteristics.

A. Manipulating Timbre during Inference

To minimize the timbre-wise domain mismatch between training and test data, we apply EQ normalization [35], [36] so that inputs at the inference stage can mimic the average frequency response of training data. This experiment is based on two hypotheses: *i)* separation networks are able to learn the overall frequency response of collections of training examples, *ii)* such networks can achieve better performance when given an input which has closer overall frequency to training data.

We use the following experimental process: *1)* Capture a reference spectrum from a collection of training data samples, either full songs of original mixes, chunks of original mixes, or chunks of random mixes; *2)* At inference stage, calculate an FIR filter that alters the overall input spectrum to the reference spectrum, and apply that filter to the input; *3)* Process the normalized input with separation networks trained with $p = 0.0$ (random mixes) or $p = 1.0$ (original mixes); *4)* Apply the inverse filter to the outputs of the separation networks. For the detailed procedures on how to obtain and apply the filters, we follow the EQ normalization methods in [36]. Figure 8(a) depicts the average frequency curves from 86 original songs in the training dataset, and Fig. 8(b) shows their spectra after applying a Savitsky-Golay filter for smoothing [37]. In our preliminary experiments, we observed that the SDR scores of the final outputs of this procedure significantly degraded while the perceptual quality of the network outputs remained fine, because of the phase mismatch caused by the filtering. We thus apply the filter and the inverse filter to the ground-truth sources based on the reference spectrum to avoid any phase-related issues.

TABLE 2. Test SDR [dB] when using EQ normalization during inference.

p	EQ Norm	Ref. Spec.	*vocals*	*bass*	*drums*	*other*	*avg*
1.0	-	-	7.6	5.6	6.7	4.8	6.2
	✓	orig. mixes (full song)	7.5	4.6	6.6	4.6	5.8
		orig. mixes (chunks)	7.5	4.7	6.5	4.5	5.8
		rand. mixes	7.5	4.6	6.5	4.6	5.8
0.0	-	-	**9.6**	**8.1**	**8.7**	6.6	**8.2**
	✓	orig. mixes (full song)	9.5	7.5	8.4	**6.8**	8.0
		orig. mixes (chunks)	9.5	7.3	8.4	6.6	7.9
		rand. mixes	9.5	7.5	8.4	6.7	8.0

The results obtained when applying EQ normalization during inference are reported in Table 2. Both for models trained with $p = 0.0$ (random mixes) and $p = 1.0$ (original mixes), EQ normalization does not show any performance improvements except for the *other* stem of the $p = 0.0$ model. These experimental results seem to show that our hypothesis that separation networks are able to learn the timbre of training data does not hold. We consider that this may be because timbre is over-simplified in our experiments. Since timbre should reflect the momentary frequency distribution, it is difficult to say that accurately reflecting it can be achieved by matching the overall frequency to a single target spectrum. Additionally, because the TFC-TDF-UNet v3 we use handles input starting from a localized part of spectrograms rather than the entire sequence, the diversity of momentary timbre presented to the network may have been too varied. This could have made it challenging for the network to learn to perform well with any single reference spectrum. Nevertheless, it is noteworthy that EQ normalization provided a performance improvement for the *other* stem. This may be because the *other* category inherently contains

Algorithm 1 EQ Augmentation

procedure EQAUG(x)
  $\log_{10} F_{\mathrm{ls}} \sim \mathcal{U}(\log_{10} F_{\mathrm{ls}_{\min}}, \log_{10} F_{\mathrm{ls}_{\max}})$
  $\log_{10} F_{\mathrm{hs}} \sim \mathcal{U}(\log_{10} F_{\mathrm{hs}_{\min}}, \log_{10} F_{\mathrm{hs}_{\max}})$
  $Q_{\mathrm{ls}}, Q_{\mathrm{hs}} \leftarrow Q_{\min}(Q_{\max}/Q_{\min})^z, z \sim \mathcal{U}(0,1)$
  $G_{\mathrm{ls}}, G_{\mathrm{hs}} \sim \mathcal{U}(G_{\min}, G_{\max})$
  Sample $k_{\mathrm{p}} \in [0, k_{\mathrm{p}_{\max}}]$
  if k_{p} is not 0 **then**
    Divide [F_{ls}, F_{hs}] into k_{p} regions spaced evenly on a logarithmic scale, and set each region's center frequency as F_{c_k}.
    for $k \leftarrow 1$ to k_{p} **do**
      $F_k \sim \mathcal{N}(F_{\mathrm{c}_k}, F_{\mathrm{c}_k}^j)$
      $F_k \leftarrow \min\{F_{\mathrm{p}_{\max}}, \max\{F_{\mathrm{p}_{\min}}, F_k\}$
      $Q_k \leftarrow Q_{\min}(Q_{\max}/Q_{\min})^z, z \sim \mathcal{U}(0,1)$
      $G_k \sim \mathcal{U}(G_{\min}, G_{\max})$
    end for
  end if
  Apply filters for x
end procedure

a much broader range of instruments and timbral variations compared to *vocals*, *bass*, or *drums*. EQ normalization might have helped mitigate such variability by partially aligning their overall spectral characteristics, which could explain why the effect was observed only for the *other* stem. Future work could explore more detailed computational proxies (e.g., MFCCs) to better capture timbral characteristics and validate our findings.

B. EQ Augmentation: Diversifying Timbre during Training

To maximize the diversity of timbre, we propose to use an EQ augmentation technique when generating random mixes for training. Specifically, the EQ augmentation (EQAugment) can be performed on two different levels, on the stems and on the mixture. Both of them are straight-forward in terms of music production; engineers often apply EQ on stems or each instrument to control the timbre of each source, while they use EQ on mixtures to manipulate the overall tone characteristic of a whole music. In our training data generation process, we newly introduce a probability q_{ind} of applying EQ at the individual-stem level, and a probability q_{mix} of applying EQ at the mixture level, for a given batch element.

Although EQ is a fundamental signal processor to manipulate timbre and has been used as a data augmentation in a few modern signal processing literature, such as in speech enhancement [38], [39] or speech self-supervised models [40], it has been rarely used in music source separation, except for a few attempts such as [41]. We describe our method for using EQ for data augmentation when training music source separation networks in Algorithm 1. We first set the cutoff frequencies of low and high shelf filters, F_{ls} and F_{hs}, by uniformly sampling them in the intervals $[F_{\mathrm{ls}_{\min}}, F_{\mathrm{ls}_{\max}}]$ and $[F_{\mathrm{hs}_{\min}}, F_{\mathrm{hs}_{\max}}]$, respectively. Then, we randomly choose their gains (G_{ls} and G_{hs}) by uniformly sampling them in $[G_{\min}, G_{\max}]$. For peaking EQ, we sample the number of peaking filters k_{p} between 1 and $k_{\mathrm{p}_{\max}}$, and divide the frequency regions [F_{ls}, F_{hs}] into k_{p} regions spaced evenly on a logarithmic scale. It should be noted that the symbol p in this section indicates *peaking filter* parameters, and should not be confused with the symbol p used in Fig. 1, Fig. 4, and previous sections to represent the probability of sampling original mixes during training. We denote the k-th region's center frequency as F_{c_k}. For each k-th peaking filter, the cutoff frequency F_k is randomly chosen from a Gaussian distribution with mean F_{c_k} and standard deviation $F_{\mathrm{c}_k}^j$ (i.e., F_{c_k} raised to the power j, where j is a hyperparameter). Subsequently, to avoid excessively large or small cutoff frequencies, each sampled F_k is constrained within the range defined by $F_{\mathrm{p}_{\min}}$ and $F_{\mathrm{p}_{\max}}$. The quality factors Q_k, Q_{ls}, and Q_{hs} are determined randomly following [40], and the gain G_k is uniformly sampled from $[G_{\min}, G_{\max}]$. Denoting the k-th peaking EQ filter as $f^{[k]}$, $k = 1, \ldots, k_{\mathrm{p}}$, the low shelf EQ filter as $f^{[0]}$, and the high shelf EQ filter as $f^{[k_{\mathrm{p}}+1]}$, we define the resulting EQ filter as

$$\bar{f} = \prod_{k=0}^{k_{\mathrm{p}}+1} f^{[k]}. \tag{5}$$

We can randomly sample and define such filters for each mixture and/or each individual stem in a mixture. Denoting by $y^{(n)}$ the n-th stem of a mixture as in Eq. 4, a random EQ filter $\bar{f}^{(n)}$ can be applied for each individual stem as

$$y_{\mathrm{eq}}^{(n)} = \bar{f}^{(n)}(y^{(n)}), \tag{6}$$

with the probability of applying individual-stem filtering to a batch element denoted as q_{ind} (individual-stem EQ is either applied independently to all stems or to none). Similarly, a filter $\bar{f}$ can be applied, with probability q_{mix}, on a mixture y of individual stems, where these individual stems may themselves have been already applied individual EQ filters as described above:

$$y_{\mathrm{eq}} = \bar{f}(y). \tag{7}$$

TABLE 3. Hyperparameters for EQ Augmentation. The unit of frequencies (F) is [Hz].

$F_{\mathrm{ls}_{\min}}$	$F_{\mathrm{hs}_{\max}}$	$F_{\mathrm{hs}_{\min}}$	$F_{\mathrm{hs}_{\max}}$	$F_{\mathrm{p}_{\min}}$	$F_{\mathrm{p}_{\max}}$	$k_{\mathrm{p}_{\max}}$	j
50	150	6000	12000	50	10000	4	0.75

TABLE 4. Hyperparameters of EQ Augmentation for individual stems and mixture. The unit of gains (G) is [dB].

Stem				Mixture			
$Q_{\min}$	$Q_{\max}$	$G_{\min}$	$G_{\max}$	$Q_{\min}$	$Q_{\max}$	$G_{\min}$	$G_{\max}$
5.0	0.7	-9.0	9.0	3.0	0.7	-6.0	6.0

TABLE 5. Test SDR [dB] for $p = 1.0$ models (original mixes only) trained with EQ augmentation.

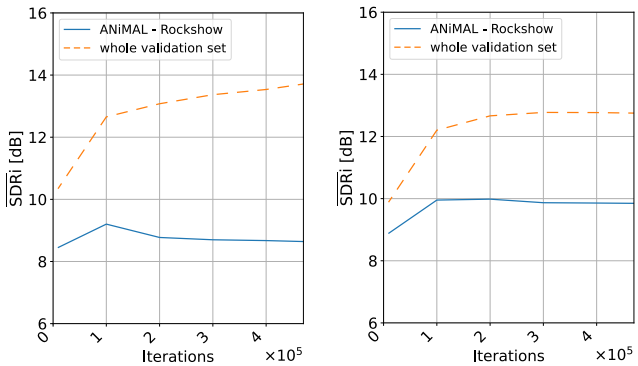
p	q_{ind}	q_{mix}	Steps	*vocals*	*bass*	*drums*	*other*	*avg*
1.0	0.0	0.0	470k	7.6	5.6	6.7	4.8	6.2
	0.5	0.5		**8.2**	**6.5**	7.7	5.4	**7.0**
	0.5	0.0		7.8	6.2	7.5	5.3	6.7
	0.0	0.5		7.7	5.5	6.8	5.0	6.3
	1.0	1.0		**8.2**	**6.5**	7.7	**5.5**	**7.0**
	1.0	0.0		7.9	**6.5**	**8.0**	5.4	**7.0**
	0	1		7.9	5.8	6.9	4.9	6.4

TABLE 6. Test SDR [dB] for $p = 0.0$ models (random mixes only) trained with EQ augmentation.

p	q_{ind}	q_{mix}	Steps	*vocals*	*bass*	*drums*	*other*	*avg*
0.0	0.0	0.0	470k	9.6	8.1	8.7	6.6	8.2
			1M	9.7	8.5	8.6	6.9	8.4
	0.5	0.5	470k	9.2	8.3	9.0	6.4	8.2
			1M	**9.9**	8.5	9.1	6.9	**8.6**
	0.5	0.0	470k	9.4	8.0	8.8	6.6	8.2
			1M	9.7	8.5	8.9	6.8	8.5
	0.0	0.5	470k	9.5	8.1	8.8	6.8	8.3
			1M	9.8	8.4	8.8	**7.0**	8.5
	1.0	1.0	470k	9.3	8.0	8.7	6.4	8.1
			1M	9.6	8.3	**9.2**	6.9	8.5
	1.0	0.0	470k	9.4	8.1	8.6	6.5	8.1
			1M	9.8	8.4	9.1	6.8	8.5
	0.0	1.0	470k	9.4	8.0	9.0	6.7	8.3
			1M	9.7	**8.6**	8.6	**7.0**	8.5

Applying EQ at the mixture level is inspired by the mix bus EQ processing, which is a widely used technique in music production to achieve better overall sound quality. In our experiments, we use the standard design of parametric EQs [42], which are made of second-order IIR filters. It should also be noted that once $\bar{f}$ is applied to a mixture, the same filter must be applied to its corresponding stems to obtain the ground-truth stems.

EQAugment can be considered as an extension of FilterAugment [43], which was originally proposed for the sound event detection task. The main difference between FilterAugment and EQAugment is that EQAugment mainly uses parametric EQ (i.e., second-order IIR filters), the fundamental tool for music production, which brings maximal timbral diversity without drastically modifying the frequency response of a given source, e.g., applying a high-pass filter with a 1 kHz cutoff frequency to a *bass* stem. Furthermore, EQAugment is performed on both the mixture and single target source levels, while [43] applied filters on a single source as the task itself did not require any combination of multiple sources. EQAugment also closely resembles the random frequency shaping method in [40], which used parametric EQ for self-supervised learning of speech. The main difference between that method and EQAugment is that the number of peaking filters and their cutoff frequencies are variable in EQAugment to maximally diversify the timbre of random mixes, while those in [40] are fixed.



(a) Plain mixes (b) Mixes w/ polyphonic bass

FIGURE 9. Average $\overline{\text{SDR}}$ during training on the validation song "ANiMAL - Rockshow", which has a polyphonic *bass*. Models were trained on (a) plain random mixes, (b) random mixes with multiple sampling of *bass* only.

We experiment with diverse combinations of q_{ind} and q_{mix}, with hyperparameters shown in Table 3 and Table 4, to validate the effectiveness of EQ augmentation in training music source separation networks, as seen in Table 5 and Table 6. We specifically use `pedalboard` [44] as our EQ, which follows the filter design of [42]. As can be seen in Table 5 and Table 6, EQ augmentation is helpful in both $p = 1.0$ and $p = 0.0$ models. Specifically, the performance gains are more significant in $p = 1.0$ models (up to 0.8 dB for *avg*) where random mixing is absent during training. For $p = 0.0$ models, the gains are negligible in case of models trained 470k steps (0.1 dB at most). We conjectured that EQ augmentation introduced more diversity of training data and would likely require more training steps to be effective. We thus kept training the models until 1M steps. As expected, the performance improvements between 470k and 1M steps are slightly larger when using EQ augmentation (0.4 dB for *avg* when $q_{\text{ind}} = 0.5$, $q_{\text{mix}} = 0.5$) compared to the baseline (0.2 dB for *avg* when $q_{\text{ind}} = 0.0$, $q_{\text{mix}} = 0.0$).

VI. Making Random Mixes More Random

A. Motivation: A Polyphonic Bass Example

Checking the training dynamics of each individual training and validation songs as in Section III.B, we found that all of the examples had consistent trends of gradually increasing performance across the training, except for one outlier, "ANiMAL - Rockshow", which had a gradually decreasing performance as shown in Fig. 9(a). We further investigated

TABLE 7. Test SDR [dB] of models trained using random mixes where some of the target sources are obtained by multiple sampling.

	Number of Samples per Stem									
System Name	*vocals*	*bass*	*drums*	*other*	p_{ms}	*vocals*	*bass*	*drums*	*other*	*avg*
Baseline	1	1	1	1	-	**9.6**	8.1	8.7	6.6	8.2
A-2	2	2	2	2	1.0	9.4	7.1	7.8	6.0	7.6
					0.5	**9.6**	7.4	8.6	6.3	8.0
					0.1	**9.6**	7.8	8.8	6.7	8.2
V-2	2	1	1	1	1.0	9.4	8.0	8.3	6.6	8.1
					0.5	**9.6**	8.3	8.6	6.7	8.3
					0.1	9.5	**8.4**	8.4	6.7	8.3
B-2	1	2	1	1	1.0	9.3	6.4	7.4	6.1	7.3
					0.5	9.5	7.1	8.1	6.4	7.7
					0.1	9.5	7.8	8.2	6.7	8.1
D-2	1	1	2	1	1.0	9.5	8.3	8.8	6.7	8.3
					0.5	9.5	8.3	**9.2**	6.7	**8.4**
					0.1	**9.6**	8.3	9.0	**6.8**	**8.4**
O-2	1	1	1	2	1.0	9.3	8.3	7.7	6.1	7.8
					0.5	9.5	8.2	8.5	6.3	8.1
					0.1	9.5	8.2	8.4	6.4	8.1

this example by listening to its stems and found that it includes a polyphonic bass, which is a quite rare case in MUSDB18-HQ as well as popular music. Notice that this outlier can significantly decrease the average performance across 4 sources because it not only decreases the score on the corresponding target source (i.e., bass), but also decreases the score on the remaining sources since the leakage of bass will be contaminate the other sources. To alleviate this issue, we experiment with a method where we sample multiple examples to make a single target source.

B. Multiple Sampling of Target Sources

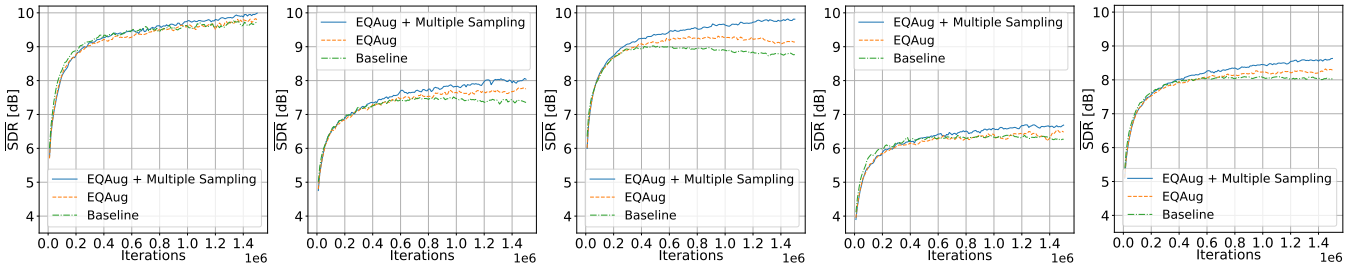
The method consisting in sampling each target source multiple times was previously proposed under the name *mix-audio* augmentation [45]. The intuition behind this technique is simple; say, if we sample *drums* two times and mix them, this mixed source can still be considered as a *drums* target. Furthermore, these mixed stems result in more data diversity even though they may sound unnatural. This sampling technique was also shown to be effective for *vocals*, but on average, it performed worse than plain random mixing, i.e., sampling each target source only once (see Table 2 in [45]).

Here, we further analyze this technique by varying the number of times we sample each stem. For example, it is possible to consider sampling only *drums* twice, instead of sampling all source stems twice, as originally proposed in [45]. Since multiple sampling can increase the diversity of random mixes while simultaneously amplifying their unnaturalness (i.e., causing a "stem-level domain shift" that may impede learning), we hypothesize that there exists a trade-off between such factors. Furthermore, we introduce a probability of multiple sampling p_{ms} in making training batches so that models can benefit from both plain and multi-sampled random mixes.

In Table 7, we note that sampling all target sources twice (A-2, $p_{ms} = 1$) results in $0.6\,\mathrm{dB}$ SDR decrease on average, and unlike [45], the performance on *vocals* also decreases by $0.2\,\mathrm{dB}$. On the other hand, sampling only one source twice leads to less performance decrease, and sampling *drums* twice (D-2, $p_{ms} = 1$) even results in slight performance gain ($0.1\,\mathrm{dB}$). Furthermore, when we use both plain random mixing and multiple sampling with $p_{ms} = 0.5$ or 0.1, we observe performance gain compared to $p_{ms} = 1$ in every case.

We hypothesize that using target sources containing multiple samples of instruments besides *drums* does not benefit performance because the additional frequency overlap caused by multiple sampling may change the timbre of the target source, which complicates the learning process. On the other hand, the percussive characteristic of *drums* means that mixing multiple *drums* tracks may change the temporal density of drum strikes, which increases data diversity while not being too different from the original mixes in terms of timbre.

Back to the "ANiMAL - Rockshow", which had a polyphonic *bass*, we check whether sampling *bass* twice actually solves the issue discussed in Section VI.A. Interestingly, as shown in solid blue curves in Fig. 9, it is effective indeed, resulting in $1.2\,\mathrm{dB}$ performance gain on average $\overline{\mathrm{SDR}}$ for that song at the final 470k training iterations. However, this also causes a performance degradation on the whole validation set (dashed orange curves in Fig. 9) and test set (Table 7). Thus,



(a) Vocals (b) Bass (c) Drums (d) Other (e) Avg

FIGURE 10. **Test $\overline{\text{SDR}}$ of models across iterations until 1.5M training steps. Models are trained with and without EQ augmentation and multiple sampling of *drums* target source. Green dash-dotted lines depict the baseline performances (without multiple sampling and EQ augmentation). Orange dashed lines depict the models trained with EQ augmentation and without multiple sampling. Blue solid lines depict the models trained with both multiple sampling and EQ augmentation.**

it seems important to consider real-world use cases before training networks; if one would like to use a MSS model for electronic music that has complex polyphonic *bass*, sampling *bass* twice may be a good strategy, however, for most of other use cases, it may not.

VII. Improving Random Mixing

In previous sections, we observed that the methods of sampling and mixing target examples multiple times (multiple sampling), and applying EQ for diversifying timbre while generating random mixes are helpful to improve music source separation networks. In this section, we train models using both multiple sampling and EQ Augmentation methods for further improvements of music source separation networks.

A. Training Longer with Multiple Sampling and EQAugment

Here we compare the models trained with or without multiple sampling and EQ augmentation. Specifically, we train 3 models for 1.5M training steps: a baseline model trained without multiple sampling and EQAugment, a model trained with EQAugment and without multiple sampling, and a model trained with multiple sampling of *drums* and EQAugmentation.

Until 3-400k training steps, all of the models show similar performance and test curves on 4 stems as seen in Fig. 10. After 400k steps, the baseline scores stop increasing except for *vocals*, while scores of the other models keep increasing. The model with EQ augmentation (orange curves in Fig. 10) shows especially better performance than the baseline on *bass* and *drums*. However, after 1M steps, the score on *drums* starts decreasing. Using multiple sampling of *drums* seems to be beneficial for this case. As shown by the blue curves in Fig. 10, the model with both multiple sampling and EQ augmentation especially shows higher performances than the model only trained with EQ augmentation not only on *drums* but also on *bass* and *other*. Although we did not train this model any longer than 1.5M steps, there seems to be room for further performance gain with more training steps.

TABLE 8. **Test SDR [dB] for models trained with random mixes ($p = 0.0$) obtained using EQ augmentation ($q_{\text{ind}} = 0.5$ and $q_{\text{mix}} = 0.5$) and multiple sampling of *drums* target source ($p_{ms} = 0.5$). ES indicates early stopping.**

Model	Multiple Sampling	EQ Aug.	Training Steps	*vocals*	*bass*	*drums*	*other*	*avg*
Open -Unmix	-	-	ES	6.1	5.4	5.8	3.9	5.3
	-	✓	ES	6.2	5.2	5.7	4.1	5.3
	✓	✓	ES	5.8	5.2	6.2	3.8	5.3
TFC -TDF -UNet v3	-	-	470k	9.6	8.1	8.7	6.6	8.2
			1M	9.7	8.5	8.6	6.9	8.4
			1.5M	9.9	8.4	8.6	6.9	8.5
	-	✓	470k	9.2	8.3	9.0	6.4	8.2
			1M	9.9	8.5	9.1	6.9	8.6
			1.5M	9.9	8.6	8.9	7.1	8.6
	✓	✓	470k	9.5	8.4	9.2	6.6	8.4
			1M	9.7	8.8	9.7	6.9	8.8
			1.5M	**10.1**	**9.1**	**9.8**	**7.1**	**9.0**

B. Effectiveness on Different Architectures

We also check the effectiveness of multiple sampling and EQ augmentation on an architecture other than TFC-TDF-UNet v3. We focus on the Open-Unmix [1] architecture for our experiments, and follow the training configurations of its official implementation including early stopping (ES in Table 8) as described in Section II.B. Other high-quality open-source separation networks such as Demucs [46] were not considered because of their high computational requirements.

As seen in Table 8, using EQ augmentation and multiple sampling does not lead to any performance gain compared to the Open-Unmix baseline, while it leads to a 0.5 dB gain on *avg* at 1.5M training steps for TFC-TDF-UNet

TABLE 9. Test SDR [dB] comparisons between recent music source separation models and our proposed method.

Model	Extra Data [h]	*vocals*	*bass*	*drums*	*other*	*avg*
HT-Demucs	-	7.9	8.5	7.9	5.7	7.5
BS-RNN	-	10.0	7.2	9.0	6.7	8.2
BS-RoFormer	-	**10.7**	**11.3**	9.5	**7.7**	**9.8**
TFC-TDF-UNet v3	-	9.9	8.4	8.6	6.9	8.5
Ours (TFC-TDF-UNet v3 w/ multiple sampling, EQAug)	-	10.1	9.1	**9.8**	7.1	9.0
HT-Demucs	800	8.9	9.8	10.1	6.4	8.8
BS-RNN	1750	10.5	8.2	10.2	7.1	9.0
BS-RoFormer	500	**12.7**	**13.3**	**12.9**	**9.0**	**12.0**

v3. This implies that these techniques may not necessarily be effective for all types of separation networks. As we already observed in Section III.D, we suspect this behavior heavily depends on the size of the network; since Open-Unmix consists of 5 times less trainable network parameters (14M) than TFC-TDF-UNet v3 (70M), it is possible that its ability to take advantage of diverse data is more limited than that of larger models. Moreover, this discrepancy may also be related to differences in network architecture design. While TFC-TDF-UNet v3 is based on a convolutional U-Net structure that processes spectrograms locally and hierarchically, Open-Unmix relies on a Bi-LSTM backbone that models temporal dependencies in a more sequential manner. Such architectural differences could lead to varying sensitivities to data augmentation strategies like multiple sampling or EQ augmentation, potentially explaining the observed performance gap.

C. Comparison with State-of-the-Art Models

Finally, we compare our proposed method with recent high quality music source separation networks. As shown in Table 9, our model, which was only trained with 100 songs from MUSDB18-HQ data, shows better performance (9.0 dB on *avg*) than an HT-Demucs [46] model trained with 800 extra songs (8.8 dB). Also, our model achieves comparable performance to a BS-RNN [5] model trained with 1750 additional songs (9.0 dB). When no extra training data is available, our model achieves the second best performance after BS-RoFormer [47], which shows 9.8 dB on *avg*. Since our methods, multiple sampling *drums* and EQ augmentation, are universally applicable for training other networks that use random mixing, and the effect of data diversity seems to be more beneficial for more powerful models, we believe they hold broad potential. We conjecture that there may be even more room for improving the current state-of-the-art networks, such as BS-RoFormer [47], compared to TFC-TDF-UNet v3, as Transformer-based models [48] are generally known to be more data-hungry. However, due to limitations in computing resources, we leave this exploration for future work.

VIII. Conclusions and Future Work

In this paper, we took steps towards explaining the effectiveness of random mixing, an obviously non-musical data augmentation technique that has become indispensable in music source separation research. Our results illustrated that both data diversity and domain consistency in terms of beat and tonality are important for music source separation, but the data diversity provided by random mixing seems more beneficial unless the amount of random mixes is strictly restricted. We also empirically demonstrated that actual performance gain of random mixing comes from the diversity of songs, not solely from the infinitely many combinations of random mixes. Then, we revisited the idea of multiple sampling of target sources in random mixing and proposed an EQ augmentation to maximize the diversity of random mixes. In our experiments, with only 100 songs and the proposed methods, we achieved competitive performance to a BS-RNN model trained with 1750 additional songs. In the future, we plan to perform additional stem-level analysis and validate our methods—especially EQ augmentation—on diverse separation tasks, such as speech separation, where it may help models generalize better to real-world recording conditions with varying microphone frequency responses. In addition, applying our analysis to larger datasets, such as MoisesDB [8], would further validate the scalability and generality of the proposed findings.

ACKNOWLEDGMENT

We are grateful to Minseok Kim, the author of TFC-TDF-UNet v3 [24], for kindly helping us to reproduce the results of the scores reported in the original paper.

REFERENCES

[1] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - A reference implementation for music source separation," *J. Open Source Softw.*, vol. 4, no. 41, p. 1667, 2019.
[2] 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.
[3] N. Takahashi, N. Goswami, and Y. Mitsufuji, "MMDenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *Proc. IWAENC*, 2018, pp. 106–110.
[4] A. Défossez, "Hybrid spectrogram and waveform source separation," in *Proc. MDX Workshop*, 2021.
[5] Y. Luo and J. Yu, "Music source separation with band-split RNN," *IEEE Trans. Audio, Speech, Lang. Process.*, 2023.
[6] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "MUSDB18-HQ - an uncompressed version of MUSDB18," Dec. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3338373
[7] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *Proc. WASPAA*, Oct. 2019.
[8] I. Pereira, F. Araújo, F. Korzeniowski, and R. Vogl, "MoisesDB: A dataset for source separation beyond 4-stems," in *Proc. ISMIR*, 2023.
[9] K. Chen, G. Wichern, F. G. Germain, and J. Le Roux, "Pac-HuBERT: Self-supervised music source separation via primitive auditory clustering and hidden-unit BERT," in *Proc. ICASSP SASB*, 2023.
[10] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. ICLR*, 2018.
[11] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proc. CVPR*, 2019, pp. 6023–6032.

[12] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 2880–2894, 2020.

[13] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 10, pp. 1702–1726, 2018.

[14] J. R. Hershey, Z. Chen, and J. Le Roux, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. ICASSP*, Mar. 2016, pp. 31–35.

[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] I. Kavalerov, S. Wisdom, H. Erdogan, B. Patton, K. Wilson, J. Le Roux, and J. R. Hershey, "Universal sound separation," in *Proc. WASPAA*, Oct. 2019.

[17] D. Petermann, G. Wichern, Z.-Q. Wang, and J. Le Roux, "The cocktail fork problem: Three-stem audio separation for real-world soundtracks," in *Proc. ICASSP*, May 2021.

[18] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *Proc. ICASSP*, 2017, pp. 261–265.

[19] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *Proc. LVA*, 2018, pp. 293–305.

[20] 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," *Front. Signal Process.*, vol. 1, 2022.

[21] G. Fabbro, S. Uhlich, C.-H. Lai, W. Choi, M. Martínez-Ramírez, W. Liao, I. Gadelha, G. Ramos, E. Hsu, H. Rodrigues *et al.*, "The Sound Demixing Challenge 2023–music demixing track," *Trans. ISMIR*, vol. 7, no. 1, 2024.

[22] C.-B. Jeon, H. Moon, K. Choi, B. S. Chon, and K. Lee, "MedleyVox: An evaluation dataset for multiple singing voices separation," in *Proc. ICASSP*, 2023, pp. 1–5.

[23] C.-B. Jeon, G. Wichern, F. G. Germain, and J. Le Roux, "Why does music source separation benefit from cacophony?" in *Proc. ICASSP Workshops*, 2024, pp. 873–877.

[24] M. Kim and J. H. Lee, "Sound Demixing Challenge 2023 –music demixing track technical report," *arXiv preprint arXiv:2306.09382*, 2023.

[25] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM J. Control Optim.*, vol. 30, no. 4, pp. 838–855, 1992.

[26] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training," in *Proc. ICLR*, 2018.

[27] C.-B. Jeon and K. Lee, "Towards robust music source separation on loud commercial music," in *Proc. ISMIR*, 2022.

[28] A. Cohen-Hadria, A. Roebel, and G. Peeters, "Improving singing voice separation using deep U-Net and Wave-U-Net with data augmentation," in *Proc. EUSIPCO*, 2019, pp. 1–5.

[29] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 4, pp. 1462–1469, 2006.

[30] S. Sarkar, E. Benetos, and M. Sandler, "Vocal harmony separation using time-domain neural networks," in *Proc. Interspeech*, 2021.

[31] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *Proc. ICASSP*, 2023, pp. 1–5.

[32] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE." *JMLR*, vol. 9, no. 11, 2008.

[33] A. Gui, H. Gamper, S. Braun, and D. Emmanouilidou, "Adapting frechet audio distance for generative music evaluation," in *Proc. ICASSP*, 2024, pp. 1331–1335.

[34] L. McInnes, J. Healy, N. Saul, and L. Großberger, "Umap: Uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, 2018.

[35] F. G. Germain, G. J. Mysore, and T. Fujioka, "Equalization matching of speech recordings in real-world environments," in *Proc. ICASSP*, 2016, pp. 609–613.

[36] M. A. Martinez Ramirez, W. Liao, C. Nagashima, G. Fabbro, S. Uhlich, and Y. Mitsufuji, "Automatic music mixing with deep learning and out-of-domain data," in *Proc. ISMIR*, 2022.

[37] R. W. Schafer, "What is a Savitzky-Golay filter? [lecture notes]," *IEEE Signal Process. Mag.*, vol. 28, no. 4, pp. 111–117, 2011.

[38] J.-M. Valin, "A hybrid DSP/deep learning approach to real-time full-band speech enhancement," in *Proc. MMSP*, 2018, pp. 1–5.

[39] H. Schroter, A. N. Escalante-B, T. Rosenkranz, and A. Maier, "DeepFilterNet: A low complexity speech enhancement framework for full-band audio based on deep filtering," in *Proc. ICASSP*, 2022, pp. 7407–7411.

[40] H.-S. Choi, J. Lee, W. Kim, J. Lee, H. Heo, and K. Lee, "Neural analysis and synthesis: Reconstructing speech from self-supervised representations," *Proc. NeurIPS*, vol. 34, pp. 16 251–16 265, 2021.

[41] C.-Y. Chiu, W.-Y. Hsiao, Y.-C. Yeh, Y.-H. Yang, and A. W.-Y. Su, "Mixing-specific data augmentation techniques for improved blind violin/piano source separation," in *Proc. MMSP*. IEEE, 2020, pp. 1–6.

[42] R. Toy, "Audio EQ cookbook." [Online]. Available: https://www.w3.org/TR/audio-eq-cookbook/

[43] H. Nam, S.-H. Kim, and Y.-H. Park, "FilterAugment: An acoustic environmental data augmentation method," in *Proc. ICASSP*, 2022, pp. 4308–4312.

[44] P. Sobot, "Pedalboard," Jul. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.7817838

[45] X. Song, Q. Kong, X. Du, and Y. Wang, "CatNet: Music source separation system with mix-audio augmentation," *arXiv preprint arXiv:2102.09966*, 2021.

[46] S. Rouard, F. Massa, and A. Défossez, "Hybrid transformers for music source separation," in *Proc. ICASSP*. IEEE, 2023, pp. 1–5.

[47] W.-T. Lu, J.-C. Wang, Q. Kong, and Y.-N. Hung, "Music source separation with band-split RoPE transformer," in *Proc. ICASSP*, 2024, pp. 481–485.

[48] A. Vaswani, Shazeer, N. N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017.