Machine Intelligence Research
www.mi-research.net

21(1), February 2024, 55-62
DOI: 10.1007/s11633-023-1441-9

Audio Mixing Inversion via Embodied Self-supervised Learning

Haotian Zhou[1,2] Feng Yu[1,2] Xihong Wu[1,2,3]

[1]Department of AI Music and Music Information Technology, Central Conservatory of Music, Beijing 100031, China

[2]Laboratory of Music Artificial Intelligence, Laboratory of Philosophy and Social Sciences, Ministry of Education, Beijing 100031, China

[3]School of Intelligence Science and Technology, Peking University, Beijing 100871, China

Abstract: Audio mixing is a crucial part of music production. For analyzing or recreating audio mixing, it is of great importance to conduct research on estimating mixing parameters used to create mixdowns from music recordings, i.e., audio mixing inversion. However, approaches of audio mixing inversion are rarely explored. A method of estimating mixing parameters from raw tracks and a stereo mixdown via embodied self-supervised learning is presented. In this work, several commonly used audio effects including gain, pan, equalization, reverb, and compression, are taken into consideration. This method is able to learn an inference neural network that takes a stereo mixdown and the raw audio sources as input and estimate mixing parameters used to create the mixdown by iteratively sampling and training. During the sampling step, the inference network predicts a set of mixing parameters, which is sampled and fed to an audio-processing framework to generate audio data for the training step. During the training step, the same network used in the sampling step is optimized with the sampled data generated from the sampling step. This method is able to explicitly model the mixing process in an interpretable way instead of using a black-box neural network model. A set of objective measures are used for evaluation. The experimental results show that this method has better performance than current state-of-the-art methods.

Keywords: Audio mixing inversion, intelligent audio mixing, self-supervised learning, audio signal processing, deep learning.

Citation: H. Zhou, F. Yu, X. Wu. Audio mixing inversion via embodied self-supervised learning. *Machine Intelligence Research*, vol.21, no.1, pp.55–62, 2024. http://doi.org/10.1007/s11633-023-1441-9

1 Introduction

Multitrack audio mixing refers to the process in which multitrack recordings are balanced, treated, and combined into a multichannel format[1]. This is usually achieved by adjusting the loudness, timbre, dynamics and spatialization of the respective musical sources, after which audio sources are collated together to obtain a mix. It is usually the mixing engineers′ responsibility to ensure that all sonic elements of the mix are properly adjusted and in accordance with the artistic intention of the composers and/or the performers. To create an ideal mix in an efficient way, a mixing engineer will apply a set of audio processors, among which gain, pan, equalization (EQ), reverb, and dynamic range compression are most frequently used. The first four effects are often implemented as linear time-invariant processing, while dynamic range compression is nonlinear. Although over the past decades, both analog and digital processing have been used for creating mixdowns, it is now often the case that a mixing engineer uses a digital audio workstation (DAW) for this task.

Audio mixing inversion refers to the process of estimating the processing used to create a certain mixdown from music recordings. This may contribute to the digital remastering of old analog recordings, since the information about what audio effects were applied and how they are applied is very useful for remastering an old recording. Such information may be recovered by the inversion of the mixing process of the given recording. In addition to remastering, audio mixing inversion may also contribute to revealing how professionals mix and master their audio recordings, which may be helpful for other musicians′ recreations of musical recordings or teaching people how to mix.

A session created by a DAW contains most, if not all, of the information on what effects and processing are applied to multitrack audio sources, and how they are applied. Thus, without the information provided by a DAW session, audio mixing inversion can be a very difficult task. It is often the case that only the mixdown and raw tracks are available, while the DAW session used to create the mixdown is not. Even in the case in which the

Research Article
Special Issue on Artificial Intelligence for Art
Manuscript received on November 29, 2022; accepted on March 23, 2023
Recommended by Associate Editor Jian-Hua Tao
Colored figures are available in the online version at https://link.springer.com/journal/11633

original DAW session is available, simply trying to reuse the session on a DAW may not work since DAW sessions cannot be shared across different DAWs or sometimes different versions of the same DAW due to potential incompatibility or inconsistency of the software. Furthermore, the session may use specific (and often costly) effect plugins unavailable to another mixing engineer, which could make the entire session unusable if the first mixing engineer employs complicated signal chains[2]. To tackle these challenges, we propose a method that utilizes an embodied self-supervised learning framework to estimate the parameters used to control the audio processors of a given DAW, and thus, to some extent if not completely, recover the processing used to create a certain mixdown from only the mixdown itself and its respective raw tracks.

2 Background

While the field of musical information retrieval has been receiving increasing attention, the task of audio mixing inversion still remains rarely explored. Early works paid much attention to recovering an audio signal in a blind manner from audio recordings with certain audio processing applied. Such problems include denoising[3], dereverberation[4], and source separation[5, 6]. Gorlow and Marchand[7] proposed an approach that uses a two-stage cascaded encoder and decoder to estimate the compression gain, and panning effects with only the spectrogram of the target mixdown. This approach requires the number of audio tracks to be fixed and does not take EQ or reverb into consideration. Gorlow and Reiss[8] proposed a method to estimate dynamic range compression applied to an audio source with satisfying results, but it can only explicitly estimate one audio signal and needs to be reformulated to model another one. More recent works have paid much attention to using black-box neural network approaches, such as using deep learning techniques for modeling EQ[9], compression[10, 11], and other effects[12]. Martínez-Ramírez[13] proposed a black-box approach of EQ and reverberation modelling. This approach is able to model EQ and reverb in an end-to-end fashion with an adaptive front-end neural network composed of 1-dimensional (1D) convolutional layers, a latent space modelling module composed of a WaveNet block and a fully connected layer, and a synthetic back-end composed of several 1-dimensional de-convolutional layers. This approach achieved state-of-the-art performances for modelling EQ and reverb, but is not very adaptive to music pieces that are not included in the dataset. Additionally, since it is a black-box approach, it cannot give out human-interpretable parameters that explicitly describe how the audio effects are used. The works mentioned above have satisfactory performances, but all of them are unable to jointly model multiple audio effects while giving out human-interpretable results and thus are inadequate for the task of audio mixing inversion.

Barchiesi and Reiss[14] proposed an approach that explicitly deals with audio mixing inversion. Their approach is able to estimate linear and nonlinear processing applied to audio tracks with both the mixdown and raw multitrack audios. Dynamic range compression is modelled as a time-varying gain envelope to be applied to the audio sources framewise, and the linear processors including EQ, gain, delay and pan are modeled by a time domain least square approach. Colonel and Reiss[2] later proposed an approach on modeling only linear processing. Their approach follows the architecture of Barchiesi and Reiss′ approach, and improves the reverb module by replacing it with a new stereo reverb model, which was achieved via a differentiable audio effect modelling framework called differentiable digital signal processing (DDSP)[15]. With the help of DDSP and deep learning techniques, their approach is able to achieve explicit white-box neural network modelling of linear audio effects and thus produces an audio processing signal chain that is interpretable to humans. Colonel et al. later applied a similar approach towards distortion[16] and dynamic range compressor[17] effects. However, although these approaches are able to produce human-interpretable results, the audio effects produced by DDSP somewhat lack quality and authenticity compared with those produced by real DAWs.

We now summarize existing audio mixing inversion methods. There are some existing works[3, 4, 7, 8] using real digital audio processors and trying to fit the target audio by finding the optimum parameters controlling the audio processors. They are able to give out human-interpretable results while maintaining quality and authenticity of the sound since they used real audio processors, however, they can only specifically model a target audio and need to be reoptimized when trying to model another audio, indicating a lack of adaptive capability. Black-box neural network based approaches[9–11, 13] are able to adapt to other real-world audios, but fail to give out human-interpretable results and the output audios sometimes lack quality and authenticity compared to real audios. DDSP-based approaches[16, 17] are adaptive to various real-world audios, and are able to give out human-interpretable parameters, but the output sound produced via DDSP somewhat lacks quality and authenticity.

Thus, we can conclude that the audio mixing inversion problem is facing 3 major challenges: 1) The lack of quality and authenticity of the output sound; 2) The ability of giving out human-interpretable results; 3) The capability of adapting to real-world audios. As mentioned above, white-box approaches lack adapting capability and black-box approaches lack interpretability and sound quality, so an obvious approach is to combine the advantages of these two kinds of approaches. DDSP somehow manages to do this, but still need to improve on sound

quality and authenticity. Another option is to replace DDSP with real digital audio processors, but the processors are not differentiable and thus are unable to coalesce with neural networks via traditional supervised learning techniques.

To tackle these challenges, we propose an embodied self-supervised learning method that can incorporate a given digital audio processing module (e.g., a DAW simulator) into the neural network optimizing procedure. Our method takes a mixdown and raw tracks as input, and estimates the parameters used for controlling the audio processors of the given processing module. Frequently used audio processors including gain, pan, EQ, reverb and dynamic range compression are taken into consideration.

This paper is organized as follows. In Section 3, the problem scenario and the model architecture as well as the optimizing procedure are demonstrated. In Section 4, details of experimental setups are shown. In Section 5, experimental results are presented and discussed. In Section 6, we conclude our work and discuss remarks and possible directions of future work.

3 Method and theory

3.1 Problem scenario

Let $\hat{y}_l(n)$ and $\hat{y}_r(n)$ denote the left and right channel of a stereo mixdown, and $y_l(n)$ and $y_r(n)$ denote the left and right channel of the stereo mixdown produced by an audio processing chain. The goal is to minimize the value of $\|y_l(n) - \hat{y}_l(n)\| + \|y_r(n) - \hat{y}_r(n)\|$. Note that $\|\cdot\|$ in the expression represents the calculation of a loss function of our choice, which will be demonstrated later.

As shown in Fig. 1, the audio signal processing chain applied to each audio track is defined as follows: gain → EQ → compressor → pan → reverb and wet/dry mix → sum with other stems. Here, the term "stem" refers to a raw track to which has been applied certain audio processing.

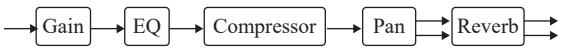

Fig. 1 Audio signal processing chain

3.2 Embodied self-supervised learning framework

We propose an embodied self-supervised learning framework for audio mixing inversion, which is inspired by the EmJEm framework[18]. EmJEm was originally designed for estimating articulatory information for a given physical simulator to generate speech. The system takes speech audio signals as input, and returns articulatory parameters used to control the physical simulator as output. This method was proposed to tackle two major challenges: Incorporating non-differentiable physical processes and the lack of paired data for supervised learning.

We use a similar iterative sampling and training strategy for our task. At the sampling step, given a set of raw audio tracks, a deep neural network is used to approximate the parameters used for controlling the DAW. Then, these parameters are sampled via Gaussian sampling with $\sigma = \sigma_{init}\gamma^t$, where σ denotes the standard deviation, σ_{init} denotes the initial value of σ, γ denotes the decay rate, and t denotes the number of current iteration. Sampled parameters are then fed into the DAW along with the raw tracks to generate a mixdown, which would then form paired data with the sampled parameters. At the training step, the same network is trained on the sampled paired data to predict mixing parameters. These two steps are operated iteratively and contribute to each other, which is somewhat similar to the guess-try-feedback process in human learning. By integrating the generation process via DAW into the learning procedure, this method is able to obtain human-interpretable parameters with which to control DAW to generate audio in a self-supervised manner.

The entire learning procedure is shown in Fig. 2. At the sampling step, we first extract the mel-scale spectrograms from the reference mixdown and its respective raw tracks as input features, and feed them into the inference network to estimate the parameters, then the parameters are sampled using multivariate Gaussian sampling and fed into the DAW to generate mixdown. At this point, the sampled parameters and the generated mixdowns form a set of paired data that can be used for supervised training in the next step. At the training step, the inference network will be trained with the sampled paired data. It should be noted that those two steps use exactly the same inference network.

The inference network is illustrated in Fig. 3. An encoder consisting of 6 U-Net[19] layers is used to obtain a feature map that contains local temporal and spectral features with different time and frequency scales. All kernel sizes of the convolutional layers are set to 3. The output dimension of the U-Net module is $N \times 80$, where N denotes the number of audio frames and 80 denotes the dimension of frequent bins. The output of the U-Net module is then mapped to an $N \times 128$ feature map with a pointwise 1D convolutional layer. For each set of input tracks, the process of estimating parameters for each track is independent of each other on a per-track basis. As a result, the cross-channel interactions, which are crucial for mixing, will not be captured by default. For this issue, we adopt a strategy that is similar to the one used in [20]. We concatenate the feature map of the target mixdown extracted by U-Net module and the point-wise 1D convolutional layer with the feature map of each

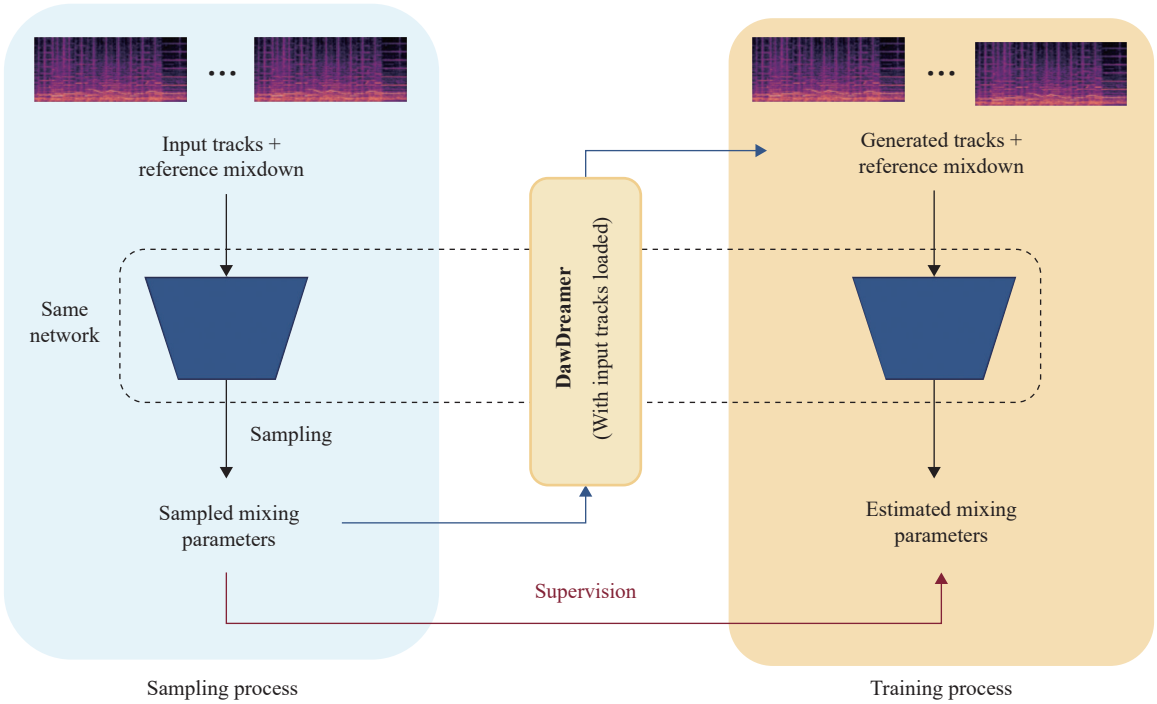

Fig. 2 Overview of an iteration. At the sampling step, the output of the inference network is sampled and used to control DawDreamer to generate audio; At the training step, the same inference network is trained with the paired data generated from the sampling step.

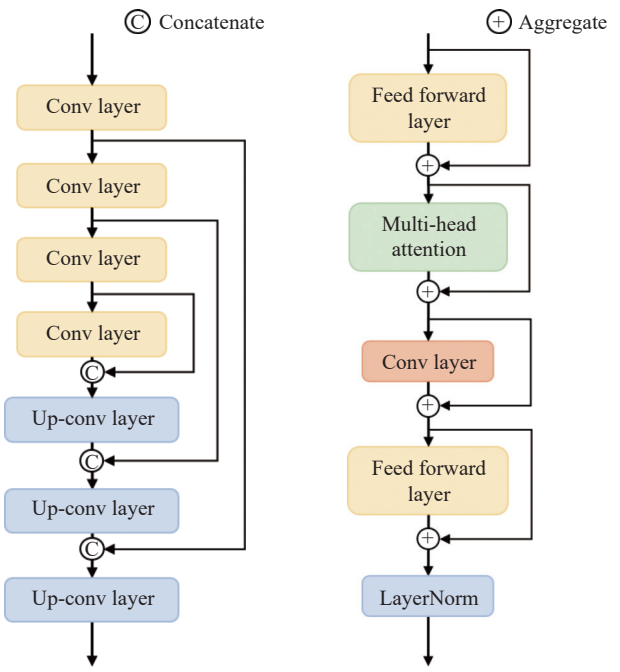

Fig. 3 Architecture of U-Net block (left) and Conformer block (right)

track, and then feed it into a Conformer module with 3 stacked Conformer[21] layers for modelling long range temporal dependencies. Each Conformer layer contains four 36-dimensional attention heads, and the kernel size is 32. The output of the Conformer module will then go through 2 bidirectional long short-term memory (LSTM) layers and a fully connected layer with Tanh activation function. We use the mean square error between the estimated parameters and the sampled parameters as the loss function for training. The entire inference network architecture is depicted as Fig. 4.

The output of the entire inference network is a 16-dimensional vector, which is used to control the DawDreamer[22] to generate mixdowns from raw multitrack audios. DawDreamer is an open source Python module that allows automated control over audio effects and processors via Python scripts. Users can compose graphs that define the workflow of multiple audio processors and set parameter automation in the graphs, serving the role of traditional DAW.

4 Experimental setup

4.1 Dataset

We conduct our tests on 2 datasets: ENST-Drums[23] and MedleyDB[24, 25]. ENST-Drums includes approximately 3 hours of multitrack drum recordings. Each item in the dataset includes 8 sources from different parts of a drum kit. We randomly split the dataset into training, validation and testing sets with the ratio of 8:1:1. The test conducted on this dataset reveals the ability of our method to perform audio mixing inversion with consistent sources and mixing techniques. Then, we conducted our experiments on the MedleyDB dataset, which

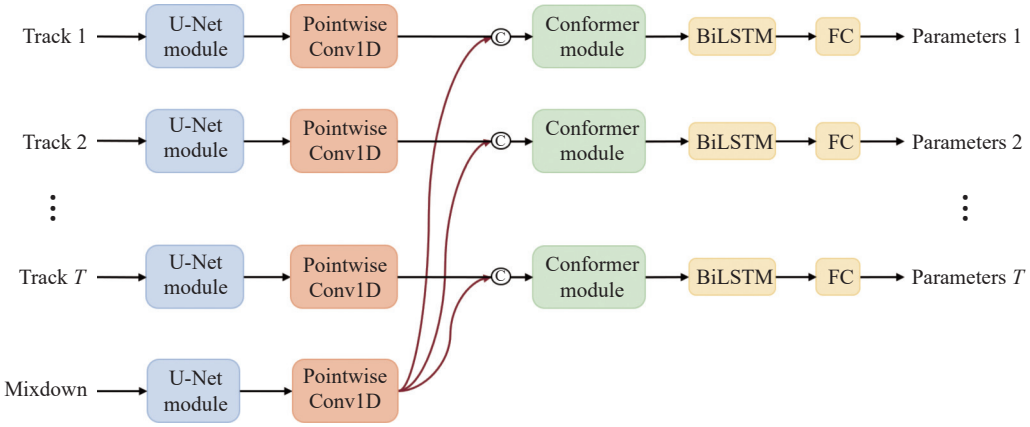

Fig. 4 Inference network architecture

provides multitrack recordings of complete songs of variant genres, sources and mixing techniques. For each song in the dataset, the raw audio tracks and the mixdown are both provided. The dataset contains 196 songs (approximately 7 h long in total). Due to memory constraints, we only use songs with no more than 16 inputs to train our model, so only 120 songs are used for our task. We randomly split the dataset into training, validation and testing sets at a ratio of 8 : 1 : 1.

4.2 Training details

For audio feature extraction, we use log magnitude mel-scale spectrogram with 80 frequency bins within the range from 20 Hz to 20 000 Hz, so the shape of a spectrogram is $N \times 80$, where N is the total frame number. Frame length and frame shift are set to 50 ms and 25 ms, respectively. All inferred mixing parameters are rescaled before being fed into the DawDreamer. The inference network is randomly initialized and the loss weight λ is set to 0.01. We use the Adam[26] optimizer with learning rate of 5×10^{-4}, $\beta_1 = 0.5$ and $\beta_2 = 0.999$ to train our inference network. For each single iteration of sampling and training, the model is trained for 10 epochs using a batch size of 4. We trained our model for 50 iterations on ENST-Drums and 100 iterations on MedleyDB. We trained our model on 2 NVIDIA 2080Ti graphic cards and the whole training process took approximately 10 hours for ENST-Drums and 51 hours for MedleyDB.

5 Results and discussion

5.1 Performance evaluation

To evaluate our method, we need to measure how close the estimated mixdowns and the reference mixdowns match. To this end, we calculate and compare the mean average error (MAE) and the multi-resolution short-term Fourier transform (STFT) distance[27] between the ground truth and predicted waveforms, and calculate the average of these 2 metrics across the dataset.

The MAE is expressed as follows:

$$MAE(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(|y_L(i) - \hat{y}_L(i)| + |y_R(i) - \hat{y}_R(i)|) \quad (1)$$

where i denotes the i-th sample, N denotes the total length of the signal, y and $\hat{y}$ denote the predicted and target signals, respectively.

The multi-resolution STFT distance is composed of a spectral convergence term θ_{sc} and a spectral log-magnitude term θ_{sm}, and is formulated as follows:

$$\theta_{sc}(y, \hat{y}) = \frac{\|STFT(y_L) - STFT(\hat{y}_L)\|_F}{\|STFT(y_L)\|_F} + \frac{\|STFT(y_R) - STFT(\hat{y}_R)\|_F}{\|STFT(y_R)\|_F} \quad (2)$$

$$\theta_{sm}(y, \hat{y}) = \frac{1}{N}(\|\log(|STFT(y_L)|) - \log(|STFT(\hat{y}_L)|)\|_{L1} + \|\log(|STFT(y_R)|) - \log(|STFT(\hat{y}_R)|)\|_{L1}) \quad (3)$$

where $\|STFT(\cdot)\|$ is the short-time Fourier transform magnitude, $\|\cdot\|_F$ is the Frobenius norm, and $\|\cdot\|_{L1}$ is the $L1$ norm. N denotes the number of STFT frames, y and $\hat{y}$ denote the predicted and target signals, respectively. Finally, we calculate M different resolutions STFTs with varying window, hopping, and frame sizes, and the error at these resolutions is averaged:

$$MRSD(y, \hat{y}) = \frac{1}{M}\sum_{m=1}^{M}(\theta_{sc}(\hat{y}, y) + \theta_{sm}(\hat{y}, y)). \quad (4)$$

The test results on ENST-Drums and MedleyDB are shown in Fig. 1. In addition to our method, we also con-

Table 1 Test results on ENST-Drums and MedleyDB

Model	ENST-Drums		MedleyDB	
	MAE	MRSD	MAE	MRSD
EAMI	**0.033**	**2.274**	**0.082**	**3.423**
Colonel and Reiss[2]	0.041	2.435	0.091	4.281
Barchiesi and Reiss[14]	0.063	2.838	0.122	5.723
Sum	0.158	3.458	0.226	7.577
Random	0.323	5.124	0.381	11.127

Note: EAMI refers to our proposed method.

sidered 4 baselines. The first one simply sums all tracks together and applies no processing. The second one uses random parameters to process each track. The third one is based on Colonel et al.'s approaches[2, 16, 17]. The fourth one is based on Barchiesi and Reiss's method[14]. For the tests for all comparing methods, we randomly split the dataset into training, validation and testing sets with the ratio of 8 : 1 : 1, which is the same as the test for our proposed method. All test samples were subjected to the same loudness normalization using Pyloudnorm[28].

We can see from the results as shown in Table 1 that our method performs significantly better than the sum and random baselines on both datasets, which proves the effectiveness of our method to approximate the reference mixdown, and our method outperforms the current state-of-the-art methods. We can also see that there is a notable difference between the results of ENST-Drums and MedleyDB, which is probably due to the different complexity of the two datasets. ENST-Drums contains only drum recordings that have the same instrument setup, while MedleyDB contains different genres of music recordings with variant sound sources including voices, synthesizers and various kinds of musical instruments. The cross-channel interactions between different tracks occur much more often in MedleyDB than ENST-Drums, which poses a significant challenge for modelling the mixing process. The difference between the performances of the method on two different datasets shows that our model suffers from the cross-channel interactions between different tracks. For example, multiple simultaneously played instruments may interfere with each other thus adding difficulty to extracting spectral features and estimating the EQ parameters. Possible improvements include introducing instrumental information and/or source separation information so that the network might be able to capture the spectral feature of each track more accurately.

5.2 Ablation study

To further prove the effectiveness of our method, we also conducted an ablation study that aims at proving the effectiveness of the proposed sampling strategy used to generate training samples during the sampling process of the proposed framework. The experiment compares the results of sampling with several different standard deviations as well as a random sampling method that is not based on the estimation of the network but in a totally random manner. The experiment was conducted only on MedleyDB dataset. We tested 6 different sampling settings: no sampling, random sampling, $\sigma = 0.05$, $\sigma = 0.1$, $\sigma = 0.5$, and σ exponentially decreasing which is described in Table 2. For each of these sampling settings, we run a test respectively. The test results are presented in Section 2. We can conclude from the result that the sampling process of the network output is necessary, since training without sampling has a significantly worse performance than others. Among all sampling settings, using an exponentially decreasing standard deviation has the best performance over all others, which further verifies the effectiveness of this sampling strategy.

Table 2 Results of different sampling settings

Sampling method	MAE	MRSD
No sampling	0.207	5.643
Random sampling	0.178	5.073
$\sigma = 0.05$	0.125	4.235
$\sigma = 0.1$	0.127	4.438
$\sigma = 0.5$	0.133	4.574
σ exponentially decreasing	**0.082**	**3.423**

6 Conclusions

We propose an embodied self-supervised learning method for audio mixing inversion. By integrating a DAW-like module, our method works in an analysis-by-generation procedure via iteratively sampling and training, which can be viewed as a form of embodied learning. Our method is able to obtain a human-interpretable parameters, which could be further analyzed or modified for educational or recreational purposes. Future work may include investigating the performances on different individual audio processors, genres and/or instrument setups, or creating a labeled dataset from unlabeled data with the help of expert knowledge.

Acknowledgements

This work was supported by High-grade, Precision and Advanced Discipline Construction Project of Beijing Universities, Major Projects of National Social Science Fund of China (No. 21ZD19) and Nation Culture and Tourism Technological Innovation Engineering Project of China.

Declarations of conflict of interest

The authors declared that they have no conflicts of interest to this work.

References

[1] R. Izhaki. *Mixing Audio: Concepts, Practices, and Tools*, New York, USA: Routledge, 2017.

[2] J. T. Colonel, J. Reiss. Reverse engineering of a recording mix with differentiable digital signal processing. *The Journal of the Acoustical Society of America*, vol. 150, pp. 608–619, 2021. DOI: 10.1121/10.0005622.

[3] G. S. Yu, S. É Mallat, E. Bacry. Audio denoising by time-frequency block thresholding. *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1830–1839, 2008. DOI: 10.1109/TSP.2007.912893.

[4] K. Lebart, J. M. Boucher, P. N. Denbigh. A new method based on spectral subtraction for speech dereverberation. *Acta Acustica United with Acustica*, vol. 87, no. 3, pp. 359–366, 2001.

[5] A. Belouchrani, K. Abed-Meraim, J. F. Cardoso, E. Moulines. A blind source separation technique using second-order statistics. *IEEE Transactions on Signal Processing*, vol. 45, no. 2, pp. 434–444, 1997. DOI: 10.1109/78.554307.

[6] A. Jourjine, S. Rickard, O. Yilmaz. Blind separation of disjoint orthogonal signals: Demixing n sources from 2 mixtures. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, vol. 5, pp. 2985–2988, 2000. DOI: 10.1109/ICASSP.2000.861162.

[7] S. Gorlow, S. Marchand. Reverse engineering stereo music recordings pursuing an informed two-stage approach, [Online], Available: https://hal.science/hal-00857676/document, 2013.

[8] S. Gorlow, J. D. Reiss. Model-based inversion of dynamic range compression. *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1434–1444, 2013. DOI: 10.1109/TASL.2013.2253099.

[9] M. Ramírez, J. D. Reiss. End-to-end equalization with convo-lutional neural networks, [Online], Available: https://www.dafx.de/paper-archive/2018/papers/DAFx2018_paper_27.pdf, 2018.

[10] S. Hawley, B. Colburn, S. I. Mimilakis. Profiling audio compressors with deep neural networks, [Online], Available: https://arxiv.org/abs/1905.11928, 2019.

[11] C. J. Steinmetz, J. D. Reiss. Efficient neural networks for real-time modeling of analog dynamic range compression, [Online], Available: https://arxiv.org/abs/2102.06200, 2021.

[12] M. A. M. Ramírez, J. D. Reiss. Modeling nonlinear audio effects with end-to-end deep neural networks. In *Proceedings of ICASSPI/IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, Brighton, UK, pp. 171–175, 2019. DOI: 10.1109/ICASSP.2019.8683529.

[13] M. A. Martínez-Ramírez. Deep Learning for Audio Effects Modeling, Ph. D. dissertation, Queen Mary University of London, UK, 2021.

[14] D. Barchiesi, J. Reiss. Reverse engineering of a mix. *Journal of The Audio Engineering Society*, vol. 58, vol. 7, pp. 563–576, 2010.

[15] J. H. Engel, L. Hantrakul, C. J. Gu, A. Roberts. DDSP: Differentiable digital signal processing. In *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2019.

[16] J. T. Colonel, M. Comunità, J. Reiss. Reverse engineering memoryless distortion effects with differentiable waveshapers, [Online], Available: https://www.eecs.qmul.ac.uk/~josh/documents/2022/21955.pdf, 2022.

[17] J. T. Colonel, J. D. Reiss. Approximating ballistics in a differentiable dynamic range compressor, [Online], Available: https://www.eecs.qmul.ac.uk/~josh/documents/2022/21915.pdf, 2022.

[18] Y. F. Sun, X. H. Wu. Embodied self-supervised learning by coordinated sampling and training, [Online], Available: https://arxiv.org/abs/2006.13350, 2020.

[19] O. Ronneberger, P. Fischer, T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the 18th International Conference on Medical Image Computing and Computer-assisted Intervention*, Springer, Munich, Germany, pp. 234–241, 2015. DOI: 10.1007/978-3-319-24574-4_28.

[20] C. J. Steinmetz, J. Pons, S. Pascual, J. Serrà. Automatic multitrack mixing with a differentiable mixing console of neural audio effects. In *Proceedings of ICASS/IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, Toronto, Canada, pp. 71–75, 2021. DOI: 10.1109/ICASSP39728.2021.9414364.

[21] A. Gulati, J. Qin, C. C. Chiu, N. Parmar, Y. Zhang, J. H. Yu, W. Han, S. B. Wang, Z. D. Zhang, Y. H. Wu, R. M. Pang. Conformer: Convolution-augmented transformer for speech recognition. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association*, Shanghai, China, pp. 5036–5040, 2020.

[22] D. Braun. DawDreamer: Bridging the gap between digital audio workstations and python interfaces, [Online], Available: https://arxiv.org/abs/2111.09931, 2021.

[23] O. Gillet, G. Richard. ENST-Drums: An extensive audio-visual database for drum signals processing. In *Proceedings of the 7th International Conference on Music Information Retrieval*, Victoria, Canada, pp. 156–159, 2006.

[24] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, J. P. Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, Taipei, China, pp. 155–160, 2014.

[25] R. Bittner, J. Wilkins, H. Yip, J. Bello. MedleyDB 2.0: New data and a system for sustainable data collection. In *Proceedings of International Conference on Music Information Retrieval*, New York, USA, 2016, [Online], Available: https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/08/bittner-medleydb.pdf.

[26] D. P. Kingma, J. Ba. Adam: A method for stochastic optimization, [Online], Available: https://arxiv.org/abs/1412.6980, 2014.

[27] R. Yamamoto, E. Song, J. M. Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Barcelona, Spain, pp. 6199–6203, 2020. DOI: 10.1109/ICASSP40776.2020.9053795.

[28] C. J. Steinmetz, J. Reiss. Pyloudnorm: A simple yet flexible loudness meter in python, [Online], Available: https://csteinmetz1.github.io/pyloudnorm-eval/paper/pyloudnorm_preprint.pdf, 2021.

Haotian Zhou received the B. Sc. degree in computer science from Peking University, China in 2018, and the M. Sc. degree in electronic and computer engineering from University of Rochester, USA in 2020. He is currently a Ph. D. degree candidate in AI music and music information technology at Central Conservatory of Music, China.

His research interest include audio and music signal processing, artificial intelligence for music and automatic music mixing.

E-mail: htzhou@mail.ccom.edu.cn

ORCID iD: 0009-0006-0876-3709

Feng Yu received the B. A. and M. A. degrees in conducting from Central Conservatory of Music, China in 1988 and 1991, respectively, and the highest Artist Diploma in conducting from Academy of Music Hanns Eisler Berlin, Germany in 1996. He was the past president of China National Opera House, and now is the president and a professor of Central Conservatory of Music, China.

His research interest include artificial intelligence for music, automatic music generation and human-computer interaction for music.

E-mail: yufeng@mail.ccom.edu.cn

Xihong Wu received the Ph. D. degree in computer science from Department of Radio Electronics, Peking University, China in 1995. He is currently a professor with School of Intelligence and Technology, Peking University, and with Department of AI Music and Music Information Technology, Central Conservatory of Music, China.

His research interests include artificial intelligence in music, machine perception, computational auditory scene analysis, speech signal processing and natural language processing.

E-mail: wxh@cis.pku.edu.cn (Corresponding author)

ORCID iD: 0009-0004-5236-7469