The Journal of Open Source Software

Open-Unmix - A reference implementation for audio source separation

Fabian-Robert Stöter[1], **Stefan Uhlich**[2], **Antoine Liutkus**[1], **and Yuki Mitsufuji**[3]

1 Inria and LIRMM, University of Montpellier, France **2** Sony Europe B.V., Germany **3** Sony Corporation, Japan

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC-BY).

Abstract

Music source separation is the task of decomposing music into its constitutive components, e.g., yielding separated stems for the vocals, bass, and drums. Such a separation has many applications ranging from rearranging/repurposing the stems (remixing, repanning, upmixing) to full extraction (karaoke, sample creation, audio restoration). Music separation has a long history of scientific activity as it is known to be a very challenging problem. In recent years, deep learning-based systems - for the first time - yielded high-quality separations that also lead to increased commercial interest. However, until now, no open-source implementation that achieves state-of-the-art results was available. *Open-Unmix* closes this gap by providing a reference implementation based on deep neural networks. It serves two main purposes: Firstly, to accelerate academic research as *Open-Unmix* provides implementations for the most popular deep learning frameworks, giving researchers a flexible way to reproduce results; Secondly, we provide a pre-trained model for end users and even artists to try and use source separation. Furthermore, we designed *Open-Unmix* to be one core component in an open ecosystem on music separation, where we already provide open datasets, software utilities, and open evaluation to foster reproducible research as the basis of future development.

Background

Music separation is a problem which fascinated researchers for over 50 years. This is partly because, mathematically, there exists no closed-form solution when many sources (instruments) are recorded a mono or stereo signal. To address the problem, researchers exploited further knowledge about the way the signals were recorded and mixed. A large number of these methods are centered around "classical" signal processing methods. For a more detailed overview see (Rafii et al. 2017) and (Cano et al. 2019). Many of these methods were hand-crafted and tuned to a small number of music recordings (Vincent et al. 2012; Araki et al. 2012; Ono et al. 2013). Systematic objective evaluation of these methods, however, was hardly feasible as freely available datasets did not exist at that time. In fact, for a meaningful evaluation, the ground truth separated stems are necessary. However, these are considered precious assets in the music recording industry, therefore, are usually not available because commercial music is well known to be subject to copyright protection laws.

Nonetheless, in the past five years, freely available datasets were released that enabled the development of data-driven methods thanks to some artists that choose compatible licenses like Creative Commons. Since then, progress in performance was closely linked

to the availability of more data that allowed the use of machine learning-based methods. This lead to a large performance boost similar to other audio tasks such as automatic speech recognition (ASR) where a large amount of data was available. In fact, in 2016 the speech recognition community had access to datasets with more than 10.000 hours of speech (Amodei et al. 2016). At the same time, the *MUSDB18* dataset was released (Rafii et al. 2017) which comprises 150 full-length music tracks – a total of just 10 hours of music. To date, this is still the largest freely available dataset for source separation. Nonetheless, even with this small amount of data, deep neural networks (DNNs) were not only successfully used for music separation but they are now setting the state-of-the-art in this domain as can be seen by the results of the community-based signal separation evaluation campaign (SiSEC) (Ono et al. 2015; Liutkus et al. 2017; Stöter, Liutkus, and Ito 2018).

In these challenges, the proposed systems are compared to other methods. Among the systems under test, classical signal processing based methods, were clearly outperformed by machine learning methods. However they were still useful as a *fast* and often *simple to understand* baseline. In fact, in the past, several systems and libraries were explicitly designed with these aspects in mind. In the following we will enlist a number of these reference implementations for source separation. While today there exist also some commercial systems such as *Audionamix XTRAX STEMS*, *IZOTOPE RX 7* or *AudioSourceRE* which target end-users, we considered only tools that are available as open-source software, suitable for research.

The first publicly available software for source separation was *openBlissart*, released in 2011 (Weninger, Lehmann, and Schuller 2011). It is written in C++ and accounts for the class of systems that are based on non-negative matrix factorization (NMF). In 2012, the *Flexible Audio Source Separation Toolbox (FASST)* was presented in (Ozerov, Vincent, and Bimbot 2011; Salaün et al. 2014). It is written in MATLAB/C++ and is also based on NMF methods, but additionally also includes other model-based methods. In 2016, the *untwist* library was proposed in (Roma et al. 2016). It comprises several methods, ranging from classical signal processing based methods to feed-forward neural networks. The library is written in Python 2.7. Unfortunately, it was not updated since 2 years and many methods are not tested. *Nussl* is a very recent framework, presented in (Manilow, Seetharaman, and Pardo 2018). It includes a large number of methods and generally focuses on classical signal processing methods rather than machine-learning-based techniques. It has built-in interfaces for common evaluation metrics and data sets. The library offers great modularity and a good level of abstraction. However, it also means that it is challenging for beginners who might only want to focus on changing the machine learning parts of the techniques.

The main problem with these implementations is that they do not deliver state-of-the-art results. No open-source system is available today that matches the performance of the best system proposed more than 4 years ago by (Uhlich, Giron, and Mitsufuji 2015). We believe that the lack of such a baseline has a serious negative impact on future research on source separation. Many new methods that were published in the last years, are usually compared to their own baseline implementations, thus showing relative instead of absolute performance gains, so that other researchers cannot assess if a method performs as good as state-of-the-art. Also, the lack of a common reference for the community potentially miss-guides young researchers and students that enter the field of music separation. The result of this can be observed by looking at the popularity of the above-mentioned music separation frameworks on GitHub: all of the frameworks mentioned above combined are less popular than two recent deep learning papers that were accompanied by code such as `MTG/DeepConvSep` from (Chandna et al. 2017) and `f90/Wave-U-Net` from (Stoller, Ewert, and Dixon 2018). Thus, users might be confused regarding which of these implementations can be considered state-of-the-art.

Open-Unmix

Today, a lot of new research in signal processing comes from applying machine learning to specific tasks such as music separation. With the rise of simple to use machine learning frameworks such as *Keras*, *Tensorflow*, *Pytorch* or *NNabla*, the technical challenge of developing a music separation system appears to be very low at first sight. However, the lack of domain knowledge about the specifics of music signals often results in weak performance where issues are difficult to track using learning-based algorithms. We could observe this problem in (Liutkus et al. 2017; Stöter, Liutkus, and Ito 2018), when we organized the source separation evaluation campaign. More complex deep architectures potentially underperform simpler models just because of subtle differences in pre- and postprocessing. These problems could have been discussed more systematically if the proposed methods would have been open-source. We, therefore, designed *Open-Unmix* to address these issues by relying on procedures that were verified by the community or proven to be working well by literature.

Design Choices

The design choices made for *Open-Unmix* have sought to reach two somewhat contradictory objectives. Its first aim is to have state-of-the-art performance, and its second aim is to still be easily understandable so that it could serve as a basis for research allowing improved performance in the future. In the past, many researchers faced difficulties in pre- and post-processing that could be avoided by sharing domain knowledge. The aim was hence to design a system that allows researchers to focus on:

- new representations, in which case there is a need for established separation architectures to use as a basis for evaluation.
- new architectures, which needs solid pre and post-processing pipelines.

In short, *Open-Unmix* inputs mixtures in the waveform domain and transforms them using a time-frequency representation, before applying a series of non-linear layers to predict the spectrogram of a target source. At the end of the chain, a postprocessing system gathers the estimates for all sources and combines them through a multichannel Wiener filter to obtain the separated waveforms.

The most critical aspects of the choices can be summarized by stating that the system heavily relies on **expert-knowledge**: while end-to-end systems that directly produce estimates in the waveform (time) domain are a promising research direction, they currently do not lead to a state-of-the-art performance in music separation. On the contrary, systems operating in the time-frequency (TF) domain are still observed to significantly outperform more "modern" solutions that would bypass the expert knowledge required by TF processing.

Framework specific vs. framework agnostic

Open-unmix is developed in parallel for multiple frameworks to cover the largest number of users. Currently, we support PyTorch, NNabla, and Tensorflow which together can be considered a sufficient coverage for our purpose. The *PyTorch* version will serve as the reference version due to its simplicity and modularity. Likewise, the NNabla is close to the PyTorch code. The Tensorflow version will be released later when Tensorflow 2.0 is stable and will be more production-oriented (inference).

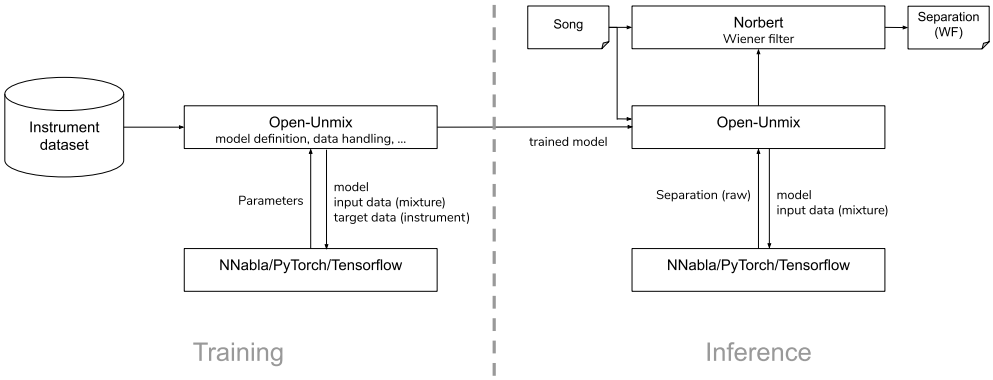

Figure 1: Block diagram of *Open-Unmix*

Hackable, Fast and Simple

Keeping in mind that the learning curve can be quite steep in audio processing, we did our best for *Open-unmix* to be:

- **simple to extend**: the pre/post-processing, data-loading, training and models part of the code is isolated and easy to replace/update. In particular, an extra-effort was done on making it easy to replace the model.
- **not a package**: keeping it easy to use to change the code made us design the software to be composed of largely independent and self-containing parts.
- **hackable (MNIST like)**: due to our objective of making it easier for machine-learning experts to try out music separation, we did our best to stick to the philosophy of baseline implementations for this community. In particular, *Open-unmix* mimics the famous MNIST example including the ability to instantly start training on data that is automatically downloaded.

Reproducible

Releasing *Open-Unmix* is first and foremost an attempt to provide a reliable implementation sticking to established programming practice as were also proposed in (McFee et al. 2018). In particular:

- **reproducible code**: everything is provided to exactly reproduce our experiments and display our results.
- **pre-trained models**: we provide pre-trained weights that allow to use the model right away or fine-tune it on user-provided data.
- **tests**: the release includes unit tests and regression tests useful to organize future open collaboration using pull-requests.

Technical Details

We will now give more technical details about *Open-Unmix*. Fig. 1 and show the basic approach. During training, we learn a DNN which can be later used for separating songs.

Datasets and Dataloaders

When designing a machine-learnig based method, our first step is to encapsulate cleanly the data-processing aspects.

- **Datasets**: we support the *MUSDB18* which is the most established dataset for music separation, that we released some years ago (Rafii et al. 2017). The dataset contains 150 full-lengths music tracks (~10h duration) of different musical styles along with their isolated `drums`, `bass`, `vocals` and `others` stems. *MUSDB18* is split into *training* (100 songs) and *test* subsets (50 songs). All files from the *MUSDB18* dataset are encoded in the Native Instruments stems format (.mp4) to reduce the file size. It is a multitrack format composed of 5 stereo streams, each one encoded in AAC @256kbps. Since AAC is bandwidth limited to 16 kHz instead of 22 kHz for full bandwidth, any model trained on *MUSDB18* would not be able to output high-quality content. As part of the release of *Open-Unmix*, we also released *MUSDB18-HQ* (Rafii et al. 2019), which is the uncompressed, full-quality version of the *MUSDB18* dataset.
- **Efficient data-loading and transforms**: since preparing the batches for training is often the efficiency bottleneck, extra-care was taken to optimize speed and performance. Here, we use a framework-specific data loading API instead of a generic module. For all frameworks we use the builtin STFT transform operator, when available, that works on the GPU to improve performance (See (Choi, Joo, and Kim 2017)).
- **Essential augmentations**: the data augmentation techniques we adopted here for source separation are described in (Uhlich et al. 2017). They enable to attain good performance even though the audio datasets such as *MUSDB18* are often of limited size.
- **Post processing**: is an important step that helps to improve the overall performance by combining the outputs of all instrument DNNs. We use a multichannel Wiener filter (MWF) as was proposed in (Nugraha, Liutkus, and Vincent 2016; Sivasankaran et al. 2015) and which we open-sourced in the `sigsep.norbert` repository

Model

General processing pipeline

The system is trained to predict a separated source from the observation of its mixture with other sources. The corresponding training is done in a *discriminative* way, i.e. through a dataset of mixtures paired with their true separated sources. These are used as ground truth targets from which gradients are computed. Although alternative ways to train a separation system have emerged recently, notably through *generative* strategies trained through adversarial cost functions, they still did not lead to comparable performance. Even if we acknowledge that such an approach could, in theory, allow scaling the size of training data since it can be done in an *unpaired* manner, we feel that this direction is still in progress and cannot be considered state-of-the-art today. That said, the *Open-Unmix* system can easily be extended to such generative training, and the community is much welcome to exploit it for that purpose.

The constitutive parts of the actual deep model used in *Open-Unmix* only comprise very classical elements, depicted in Fig. 2:

- *LSTM*: The core of *Open-Unmix* is a three-layer bidirectional LSTM network (Hochreiter and Schmidhuber 1997). Due to its recurrent nature, the model can be trained and evaluated on arbitrary length of audio signals. Since the model takes information from the past and future simultaneously, the model cannot be used in an online/real-time manner. An uni-directional model can easily be trained.
- *Fully connected time-distributed layers* are used for dimensionality reduction and augmentation, thus encoding/decoding the input and output. They allow control over the number of parameters of the model and prove to be crucial for generalization.

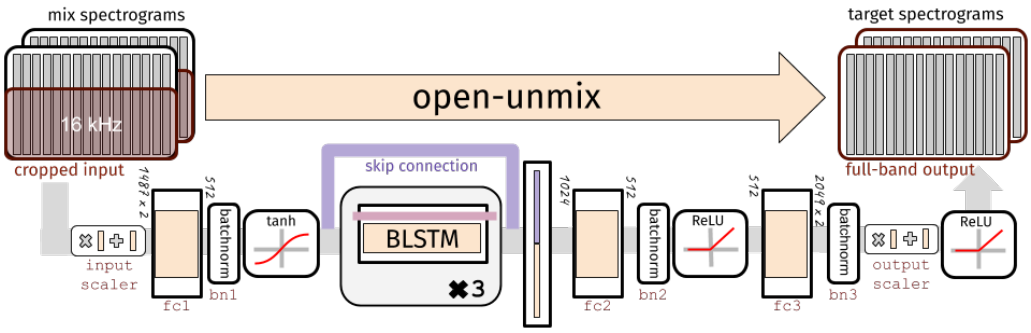

Figure 2: Separation network

- *Skip connections* are used in two ways: i/ the output to recurrent layers are augmented with their input, and this proved to help convergence. ii/ The output spectrogram is computed as an element-wise multiplication of the input. This means that the system has to learn *how much each TF bin does belong to the target source* and not the *actual* value of that bin. This is *critical* for obtaining good performance and combining the estimates given for several targets, as done in *Open-unmix*.
- *Non linearities* are of three kinds: i/ rectified linear units (ReLU) allow intermediate layers to comprise nonnegative activations, which long proved effective in TF modeling. ii/ `tanh` are known to be necessary for good training of LSTM model, notably because they avoid exploding input and output. iii/ a `sigmoid` activation is chosen before masking, to mimic the way legacy systems take the outputs as a *filtering* of the input.
- *Batch normalization* long proved important for stable training, because it makes the different batches more similar in terms of distributions. In the case of audio where signal dynamics can be very important, this is crucial.

Note that the model can process and predict multichannel spectrograms by stacking features. Furthermore, please note that the input and output to the *Open-Unmix* core deep model are magnitude spectrograms. Although using phase as additional input feature (Muth et al. 2018) or estimating the instrument phase (Le Roux et al. 2019; Takahashi et al. 2018) are interesting approaches, they have not yet been submitted to international evaluation campaigns like SiSEC for music separation.

Training

The experience gained during the research we did for releasing *Open-Unmix* taught us that successful **training** of the model requires expert knowledge that we want to share with the community, since only an implementation can enable widespread diffusion of these techniques. Indeed, those tricks are often deemed of not sufficient scientific importance to be found in scientific papers.

In particular, we use the following setup for training: We learn the weights of the BLSTM by minimizing the mean squared error (MSE) with the ADAM optimizer (Kingma and Ba 2014). We start with an initial learning rate of 0.001 which is sequentially reduced by a factor of 0.3 if the validation error plateaus. Besides saving the current model, we also save the best model on the validation dataset, i.e., perform early stopping. The validation set consists of 14 songs, which we selected from the 100 training songs. For reproducibility, the validation split is part of the (Stöter and Liutkus 2019c) tools.

Furthermore, heavy data augmentation is used due to the small size of MUSDB. We use the data augmentation as described in (Uhlich et al. 2017):

- random swapping left/right channel for each instrument,
- random scaling with uniform amplitudes from [0.25,1.25],
- random chunking into sequences for each instrument, and,
- random mixing of instruments from different songs.

For training the recurrent layers, we use sequences that correspond to six seconds duration and use 16 samples per minibatch.

As shown in Fig. 2, the model uses an input scaler and output scaler, which both subtract an offset and multiply with a scale for each frequency bin. For the input scaler, we initialize the offset and scale by the mean and standard deviation of the mixture magnitudes, which are computed from the training dataset. For the output scaler, we initialize the offset and scale to 1.0, i.e., the network is initialized such that it starts from a mask with all-ones, i.e., it uses the mixture signal as the first estimate.

Results

The final models were trained using the PyTorch version of *Open-Unmix* on the original version of *MUSDB18* but also on *MUSDB-HQ* as mentioned earlier. Both models were evaluated using museval (Stöter and Liutkus 2019a) on the test set of *MUSDB18* such that we can compare their performance to the other participants of the SiSEC 2018 contest (Stöter, Liutkus, and Ito 2018). The result scores are listed in Table 1. It is important to note that these scores are aggregated using median over the metric frames and median over the tracks. The scores in native museval JSON format, as well as the pre-trained weights, are released on zenodo [link_to_zenodo]. Furthermore, we adopted torch.hub, a system that automatically downloads pre-trained weights, thus making it very easy to use the model out of the box from python.

Concerning the performance, it is interesting to note that *UMXHQ* performs very similar to *UMX*, thus we made *UMXHQ* the default model for inference and suggest that *UMX* should only be used when compared against other participants from SiSEC 2018. The models are full stereo for input and output. A detailed list of all the parameters that were used to train the model is part of the model repository on zenodo. They also include the exact git commit that was used to train the model. [link_to_zenodo].

Table 1: BSSEval scores of *UMX* and *UMXHQ* on *MUSDB18*

target	SDR	SIR	SAR	ISR	SDR	SIR	SAR	ISR
`model`	UMX	UMX	UMX	UMX	UMXHQ	UMXHQ	UMXHQ	UMXHQ
vocals	6.32	13.33	6.52	11.93	6.25	12.95	6.50	12.70
bass	5.23	10.93	6.34	9.23	5.07	10.35	6.02	9.71
drums	5.73	11.12	6.02	10.51	6.04	11.65	5.93	11.17
other	4.02	6.59	4.74	9.31	4.28	7.10	4.62	8.78

Objective Evaluation

We compared *Open-Unmix* to other separation models that were submitted to the last SiSEC contest (Stöter, Liutkus, and Ito 2018). The results of the `UMX` are depicted in 3. It can be seen that our proposed model reaches state-of-the-art results. There is no statistically significant difference between the best method `TAK1` and `UMX`. Because `TAK1` is not released as open-source, this indicates that *Open-Unmix* is the current state-of-the-art open-source source separation systems.

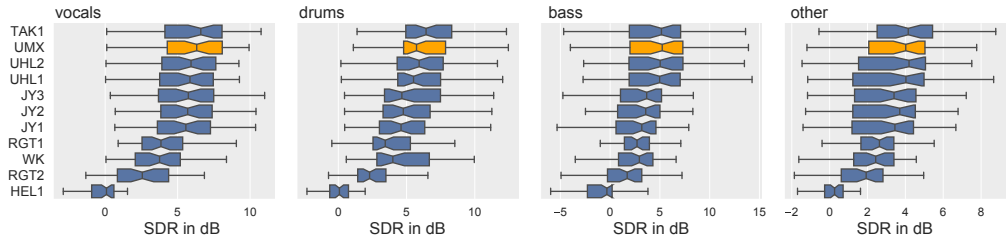

Figure 3: Boxplots of evaluation results of the `UMX` model compared with other methods from (Stöter, Liutkus, and Ito 2018) (methods that did not only use MUSDB18 for training were omitted)

Community

Open-Unmix was developed by Fabian-Robert Stöter and Antoine Liutkus at Inria Montpellier. The research concerning the deep neural network architecture as well as the training process was done in close collaboration with Stefan Uhlich and Yuki Mitsufuji from Sony Corporation.

In the future, we hope the software will be well received by the community. *Open-Unmix* is part of an ecosystem of software, datasets, and online resources: the `sigsep` community.

First, we provide MUSDB18 (Rafii et al. 2017) and MUSDB18-HQ (Rafii et al. 2019) which are the largest freely available dataset, this comes with a complete toolchain to easily parse and read the dataset (Stöter and Liutkus 2019c). We maintain *museval*, the most used evaluation package for source separation (Stöter and Liutkus 2019a). We also are the organizers of the largest source separation evaluation campaign such as (Stöter, Liutkus, and Ito 2018). And, we implemented a reference implementation of multi-channel wiener filter implementation released in (Stöter and Liutkus 2019b). The `sigsep` community is organized and presented on its own website. *Open-Unmix* itself is hosted on https://open.unmix.app, which links to the framework implementations and provide all further information such as audio demos.

Outlook

Open-Unmix is a community-focused project, we, therefore, encourage the community to submit bug-fixes and comments and improve the computational performance. However, we are not looking for changes that only focused on improving the separation performance as this would be out of scope for a baseline implementation. Instead, we expect many researchers will fork the software as a basis for their research. We prepared several custom options to easily extend the code:

1. *native dataset and data loader APIs*: This encourages the interested researcher to train *Open-unmix* model with her/his data. We, therefore, provide datasets that can easily parse random file-based data. Users of *Open-Unmix* that have their datasets and could not fit one of our predefined datasets might want to implement or use their own `torch.utils.data.Dataset` to be used for the training. Such a modification is very simple and we additionally provide a dataset template.
2. *custom models*: We think that recurrent models provide the best trade-off between good results, fast training and flexibility of training due to its ability to learn from arbitrary durations of audio and different audio representations. Furthermore, since the audio signals at test time can be of arbitrary lengths, the recurrent models yield the best consistency of the results within one audio track. If users want to try different models you can easily build on the proposed model template.

3. *joint models*: We designed *Open-Unmix* so that the training of multiple targets is handled in separate models. We think that this has several benefits such as: First, single-source models can leverage unbalanced data where for each source different size of training data is available. Second, training can easily be distributed by training multiple models on different nodes in parallel. Third, at test time the selection of different models can be adjusted for specific applications. Adjusting *Open-Unmix* to support joint training is simple, and we provide an example in our documentation.

References

Amodei, D., R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, et al. 2016. "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin." In *Icml*, 173–82.

Araki, Shoko, Francesco Nesta, Emmanuel Vincent, Zbynek Koldovsky, Guido Nolte, Andreas Ziehe, and Alexis Benichoux. 2012. "The 2011 Signal Separation Evaluation Campaign (SiSEC2011): - Audio Source Separation -." In *10th International Conference on Latent Variable Analysis and Signal Separation*.

Cano, E., D. FitzGerald, A. Liutkus, M. D. Plumbley, and F. Stöter. 2019. "Musical Source Separation: An Introduction." *IEEE Signal Processing Magazine* 36 (1):31–40.

Chandna, P., M. Miron, J. Janer, and E. Gómez. 2017. "Monoaural Audio Source Separation Using Deep Convolutional Neural Networks." In *Latent Variable Analysis and Signal Separation*, 258–66.

Choi, Keunwoo, Deokjin Joo, and Juho Kim. 2017. "Kapre: On-Gpu Audio Preprocessing Layers for a Quick Implementation of Deep Neural Network Models with Keras." In *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*. ICML.

Hochreiter, S., and J. Schmidhuber. 1997. "Long Short-Term Memory." *Neural Comput.* 9 (8). Cambridge, MA, USA: MIT Press:1735–80.

Kingma, Diederik P, and Jimmy Ba. 2014. "Adam: A Method for Stochastic Optimization." *arXiv Preprint arXiv:1412.6980*.

Le Roux, Jonathan, Gordon Wichern, Shinji Watanabe, Andy Sarroff, and John R Hershey. 2019. "Phasebook and Friends: Leveraging Discrete Representations for Source Separation." *IEEE Journal of Selected Topics in Signal Processing* 13 (2). IEEE:370–82.

Liutkus, Antoine, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. 2017. "The 2016 Signal Separation Evaluation Campaign." In *Proc. Intl. Conference on Latent Variable Analysis and Signal Separation (Lva/Ica)*, 323–32. Springer International Publishing.

Manilow, E., P. Seetharaman, and B. Pardo. 2018. "The Northwestern University Source Separation Library." In *ISMIR*, 297–305.

McFee, Brian, Jong Wook Kim, Mark Cartwright, Justin Salamon, Rachel M Bittner, and Juan Pablo Bello. 2018. "Open-Source Practices for Music Signal Processing Research: Recommendations for Transparent, Sustainable, and Reproducible Audio Research." *IEEE Signal Processing Magazine* 36 (1). IEEE:128–37.

Muth, Joachim, Stefan Uhlich, Nathanael Perraudin, Thomas Kemp, Fabien Cardinaux, and Yuki Mitsufuji. 2018. "Improving Dnn-Based Music Source Separation Using Phase Features." *arXiv Preprint arXiv:1807.02710*.

Nugraha, Aditya A., Antoine Liutkus, and Emmanuel Vincent. 2016. "Multichannel Audio Source Separation with Deep Neural Networks." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24 (9):1652–64.

Ono, Nobutaka, Zbynek Koldovsky, Shigeki Miyabe, and Nobutaka Ito. 2013. "The 2013 Signal Separation Evaluation Campaign." In *Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6.

Ono, Nobutaka, Zafar Rafii, Daichi Kitamura, Nobutaka Ito, and Antoine Liutkus. 2015. "The 2015 Signal Separation Evaluation Campaign." In *Proc. Intl. Conference on Latent Variable Analysis and Signal Separation (Lva/Ica)*. Liberec, Czech Republic,

Ozerov, Alexey, Emmanuel Vincent, and Frédéric Bimbot. 2011. "A General Flexible Framework for the Handling of Prior Information in Audio Source Separation." *IEEE Transactions on Audio, Speech, and Language Processing* 20 (4). IEEE:1118–33.

Rafii, Zafar, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. 2017. "MUSDB18, a Corpus for Audio Source Separation." https://sigsep.github.io/musdb.

Rafii, Zafar, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. 2019. "MUSDB18-Hq - an Uncompressed Version of Musdb18." https://doi.org/10.5281/zenodo.3338373.

Roma, Gerard, Emad M Grais, AJ Simpson, Iwona Sobieraj, and Mark D Plumbley. 2016. "Untwist: A New Toolbox for Audio Source Separation." In *Extended Abstracts for the Late-Breaking Demo Session of the 17th International Society for Music Information Retrieval Conference, Ismir*, 7–11.

Salaün, Yann, Emmanuel Vincent, Nancy Bertin, Nathan Souviraà-Labastie, Xabier Jaureguiberry, Dung T. Tran, and Frédéric Bimbot. 2014. "The Flexible Audio Source Separation Toolbox Version 2.0." ICASSP. https://hal.inria.fr/hal-00957412.

Sivasankaran, Sunit, Aditya Arie Nugraha, Emmanuel Vincent, Juan A Morales-Cordovilla, Siddharth Dalmia, Irina Illina, and Antoine Liutkus. 2015. "Robust Asr Using Neural Network Based Speech Enhancement and Feature Simulation." In *2015 Ieee Workshop on Automatic Speech Recognition and Understanding (Asru)*, 482–89. IEEE.

Stoller, Daniel, Sebastian Ewert, and Simon Dixon. 2018. "Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation." *arXiv Preprint arXiv:1806.03185*.

Stöter, Fabian-Robert, and Antoine Liutkus. 2019a. "sigsep/sigsep-mus-eval: v0.3.0." https://doi.org/10.5281/zenodo.3261102.

———. 2019b. "sigsep/norbert: v0.2.0." https://doi.org/10.5281/zenodo.3269749.

———. 2019c. "sigsep/sigsep-mus-db: v0.1.7." https://doi.org/10.5281/zenodo.3271451.

Stöter, Fabian-Robert, Antoine Liutkus, and Nobutaka Ito. 2018. "The 2018 Signal Separation Evaluation Campaign." In *Latent Variable Analysis and Signal Separation: 14th International Conference, Lva/Ica 2018, Surrey, Uk*, 293–305.

Takahashi, Naoya, Purvi Agrawal, Nabarun Goswami, and Yuki Mitsufuji. 2018. "PhaseNet: Discretized Phase Modeling with Deep Neural Networks for Audio Source Separation." In *Interspeech*, 2713–7.

Uhlich, S., F. Giron, and Y. Mitsufuji. 2015. "Deep Neural Network Based Instrument Extraction from Music." In *Icassp*, 2135–9.

Uhlich, S., M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji. 2017. "Improving Music Source Separation Based on Deep Neural Networks Through Data Augmentation and Network Blending." In *Icassp*. New Orleans, LA, USA.

Vincent, Emmanuel, Shoko Araki, Fabian J. Theis, Guido Nolte, Pau Bofill, Hiroshi Sawada, Alexey Ozerov, Vikrham Gowreesunker, Dominik Lutter, and Ngoc Q.K. Duong. 2012. "The Signal Separation Evaluation Campaign (2007-2010): Achievements and Remaining Challenges" 92 (8):1928–36.

Weninger, F., A. Lehmann, and B. Schuller. 2011. "OpenBliSSART: Design and Evaluation of a Research Toolkit for Blind Source Separation in Audio Recognition Tasks." In *Proc. IEEE Intl. Conf. On Acoustics, Speech and Signal Processing (Icassp)*, 1625–8. https://doi.org/10.1109/ICASSP.2011.5946809.