MDX
Music Demixing Workshop

Hybrid Spectrogram and Waveform Source Separation

Alexandre Défossez[1]

1 Facebook AI Research

In partnership with

ISMIR2021

Abstract

Source separation models either work on the spectrogram or waveform domain. In this work, we show how to perform end-to-end hybrid source separation, letting the model decide which domain is best suited for each source, and even combining both prediction. We propose a hybrid version of the Demucs architecture (Défossez et al., 2019) which won the Music Demixing Challenge 2021 organized by Sony. This architecture also comes with additional improvements, such as compressed residual branches, local attention or singular value regularization. Overall, a 1.5 dB improvement of the Signal-To-Distortion (SDR) was observed across all sources, an improvement confirmed by human subjective evaluation, with an overall quality rated at 2.83 out of 5, and absence of contamination at 3.04 (against 2.86 and 2.44 for the second ranking model submitted at the competition).

Introduction

Work on music source separation has recently focused on the task of separating 4 well defined instruments in a completely supervised manner: drums, bass, vocals and other accompaniments. Recent evaluation campaigns (F.-R. Stöter et al., 2018) have focused on this setting, relying on the standard MusDB benchmark (Rafii et al., 2017). In 2021, Sony organized the Music Demixing Challenge (MDX) (Mitsufuji et al., 2021), an online competition where separation models are evaluated on a completely new and hidden test set composed of 36 tracks.

The challenge featured a number of baselines to start from, which could be divided into two categories: spectrogram or waveform based methods. The former consists in models that are fed with the input spectrogram, either represented by its amplitude, such as Open-Unmix (F.-R. Stöter et al., 2019) and its variant CrossNet Open-Unmix (Sawata et al., 2020), or as the concatenation of its real and imaginary part (Complex-As-Channels, CAC, following (Choi et al., 2020)), such as LaSAFT (Choi et al., 2021). Similarly, the output can be either a mask on the input spectrogram, complex numbers or complex modulation of the input spectrogram (Kong et al., 2021).

On the other hand, waveform based models such as Demucs (Défossez et al., 2019) are directly fed with the raw waveform, and output the raw waveform for each of the source. Most of those methods will perform some kind of learnt time-frequency analysis in its first layers through convolutions, such as Demucs and Conv-TasNet (Luo & Mesgarani, 2019), although some will not rely at all on convolutional layers, like Dual-Path RNN (Luo et al., 2020).

Theoretically, there should be no difference between spectrogram and waveform model, in particular when considering CaC (complex as channels), which is only a linear change of base for the input and output space. However, this would only hold true in the limit of having an infinite amount of training data. With a constraint dataset, such as the 100 songs of MusDB, inductive bias can play an important role. In particular, spetrogram methods varies by more than their input and output space. For instance, with a notion of frequency, it is possible to apply convolution along frequencies, while waveform methods must use layers that are fully connected with respect to its channels. The final loss being still far from zero, there will also be artifacts in the separated audio. Different representations will lead to different artifacts, some being more noticeable for the drums and bass (phase inconsitency for spectrogram methods will make the attack sound hollow), while others are more noticeable for the vocals (vocals separated by Demucs suffer from crunchy static noise)

In this work, we extend the Demucs architecture to perform hybrid waveform/spectrogram domain source separation. The original U-Net architecture is extended to provide two parallel branches: one in the time (temporal) and one in the frequency (spectral) domain. We add extra improvements to the base architecture, namely compressed residual branches comprising dilated convolutions (Yu & Koltun, 2016), LSTM (Hochreiter & Schmidhuber, 1997) and local attention (Vaswani et al., 2017). We present the impact of those changes as measured on the MusDB benchmark and on the MDX hidden test set, as well as subjective evaluations. Hybrid Demucs ranked 1st at the MDX competition when trained only on MusDB, with 7.32 dB of SDR, and 2nd when extra training data was allowed.

Related work

There exist a number of spectrogram based music source separation architectures. Open-Unmix (F.-R. Stöter et al., 2019) is based on fully connected layers and a bi-LSTM. It uses multi-channel Wiener filtering (Nugraha et al., 2016) to reduce artifacts. While the original Open-Unmix is trained independently on each source, a multi-target version exist (Sawata et al., 2020), through a shared averaged representation layer. D3Net (Takahashi & Mitsufuji, 2020) is another architecture, based on dilated convolutions connected with dense skip connections. It is currently the most competitive spectrogram architecture, with an average SDR of 6.0 dB on MusDB. Unlike previous methods which are based on masking, LaSAFT (Choi et al., 2021) uses Complex-As-Channels (Choi et al., 2020) along with a U-Net (Ronneberger et al., 2015) architecture. It is also single-target, however its weights are shared across targets, using a weight modulation mechanism to select a specific source.

Waveform domain source separation was first explored by (Lluís et al., 2018), as well as (Jansson et al., 2017) and (Stoller et al., 2018) with Wave-U-Net. However, those methods were lagging in term of quality, almost 2 dB behind their spectrogram based competitors. Demucs (Défossez et al., 2019) was built upon Wave-U-Net, using faster strided convolutions rather than DSP based downsampling, allowing for much larger number of channels (but potentially introducing aliasing artifacts (Pons et al., 2021)), and extra Gated Linear Unit layers (Dauphin et al., 2017) and biLSTM. For the first time, waveform domain methods surpassed spectrogram ones when considering the overall SDR (6.3 dB on MusDB), although its performance is still inferior on the other and vocals sources. Conv-Tasnet (Luo & Mesgarani, 2019), a model based on masking over a learnt time-frequency representation using dilated convolutions, was also adapted to music source separation by (Défossez et al., 2019), but suffered from more artifacts and lower SDR.

To the best of our model, no other work has studied true end-to-end hybrid source separation, although other team in the MDX competition used model blending across domains as a simpler post-training alternative.

Architecture

In this Section we present the structure of Hybrid Demucs, as well as the other additions that were added to the original Demucs architecture.

Original Demucs

The original Demucs architecture (Défossez et al., 2019) is a U-Net (Ronneberger et al., 2015) encoder/decoder structure. A BiLSTM (Hochreiter & Schmidhuber, 1997) is applied between the encoder and decoder to provide long range context. The encoder and decoder have a symetric structure. Each encoder layer is composed of a convolution with a kernel size of 8, stride of 4 and doubling the number of channels (except for the first layer, which sets it to a fix value, typically 48 or 64). It is followed by a ReLU, and a so called 1x1 convolution with Gated Linear Unit activation (Dauphin et al., 2017), i.e. a convolution with a kernel size of 1, where the first half of the channels modulates the second half through a sigmoid. The 1x1 convolution double the channels, and the GLU halves them, keeping them constant overall. Symetrically, a decoder layer sums the contribution from the U-Net skip connection and the previous layer, apply a 1x1 convolution with GLU, then a transposed convolution that halves the number of channels (except for the outermost layer), with a kernel size of 8 and stride of 4, and a ReLU (except for the outermost layer). There are 6 encoder layers, and 6 decoder layers, for processing 44.1 kHz audio. In order to limit the impact of aliasing from the outermost layers, the input audio is upsampled by a factor of 2 before entering the encoder, and downsampled by a factor of 2 when leaving the decoder.

Hybrid Demucs

Overall architecture

Hybrid Demucs extends the original architecture with a multi-domain analysis and prediction capabilities. The model is composed of a temporal branch, a spectral branch, and shared layers. The temporal branch takes the input waveform and process like the standard Demucs. It contains 5 layers, which are going to reduce the number of time steps by a factor of $4^5 = 1024$. Compared with the original architecture, all ReLU activations are replaced by Gaussian Error Linear Units (GELU) (Hendrycks & Gimpel, 2016).

The spectral branch takes the spectrogram obtained from an STFT over 4096 time steps, with a hop length of 1024. Notice that the number of time steps immediately matches that of the output of the encoder of the temporal branch. In order to reduce the frequency dimension, we apply the same convolution as in the temporal branch, but along the frequency dimension. Each layer reduces by a factor of 4 the number of frequencies, except for the 5th layer, which reduces by a factor of 8. After being processed by the spectral encoder, the signal has only one “frequency” left, and the same number of channels and sample rate as the output of the temporal branch.

The temporal and spectral representations are summed before going through a shared encoder/decoder layer which further reduces by 2 the number of time steps (using a kernel size of 4). Its output serve both as the input of the temporal and spectral decoder. Hybrid Demucs has a dual U-Net structure, with the temporal and spectral branches having their respective skip connections.

The output of the spectral branch is inversed with the ISTFT, and summed with the temporal branch output, giving the final model prediction. Due to this overall design, the model is free to use whichever representation is most conveniant for different parts of the signal, even within one source, and can freely share information between the two representations. The hybrid architecture is represented on Figure 1.

Padding for easy alignment

One difficulty that arised when designing the architecture was to properly align the spectral and temporal representations for any input length. For an input length L, kernel size K, stride S and padding on each side P, the output of a convolution is of length $(L - K + 2 * P)/S + 1$. The usual choice of $P = K/2$ gives an output of size $L/S + 1$. For the STFT, we have a single convolution with $K = 4096$ and $S = 1024$. However, the extra $+1$ becomes problematic when stacking convolutions, and iterating the previous formula. Indeed, it is easy to have a mismatch between the number of time steps for the temporal and spectral representations.

In order to simplify computations, we instead pad by $P = (K - S)/2$, giving an output of L/S, so that matching the overall stride is now sufficiant to exactly match the length of the spectral and temporal representations. We apply this padding both for the STFT, and convolution layers in the temporal encoders.

Frequency-wise convolutions

In the spectral encoder/decoder, we use frequency wise convolution. Using the padding trick, we have that the number of frequency is divided by 4 with every layer. The initial number of frequency bin is 2049, but for simplicity, we drop the highest bin, giving 2048 frequency bins. The input of the 5th layer has 8 frequency bins, which we reduce to 1 with a convolution with a kernel size of 8 and no padding.

However, it has been noted that unlike the time axis, the distribution of musical audio signals is not truely invariant to translation along the frequency axis. Instruments have specific pitch range, vocals have well defined formants etc. To account for that, (Isik et al., 2020) suggest injecting an embedding of the frequency before applying the convolution. We use the same approach, with the addition that we smooth the initial embedding so that close frequencies have similar embeddings. We inject this embedding just before the second spectral encoder layer. We also investigated using specific weights for specific frequency bands in the outermost spectral branch layers.

Spectrogram representation

We investigated both with representing the spectrogram as complex numbers (Choi et al., 2020) or as amplitude spectrograms. For this second option, we use Wiener filtering (Nugraha et al., 2016). We use Open-Unmix implementation of this filtering (F.-R. Stöter et al., 2019), which uses an iterative estimation procedure. Using more iterations at evaluation time is usually optimal, but sadly doesn't work well with the hybrid approach, as changing the spectrogram output, without the waveform output being able to adapt, will drastically reduce the SDR.

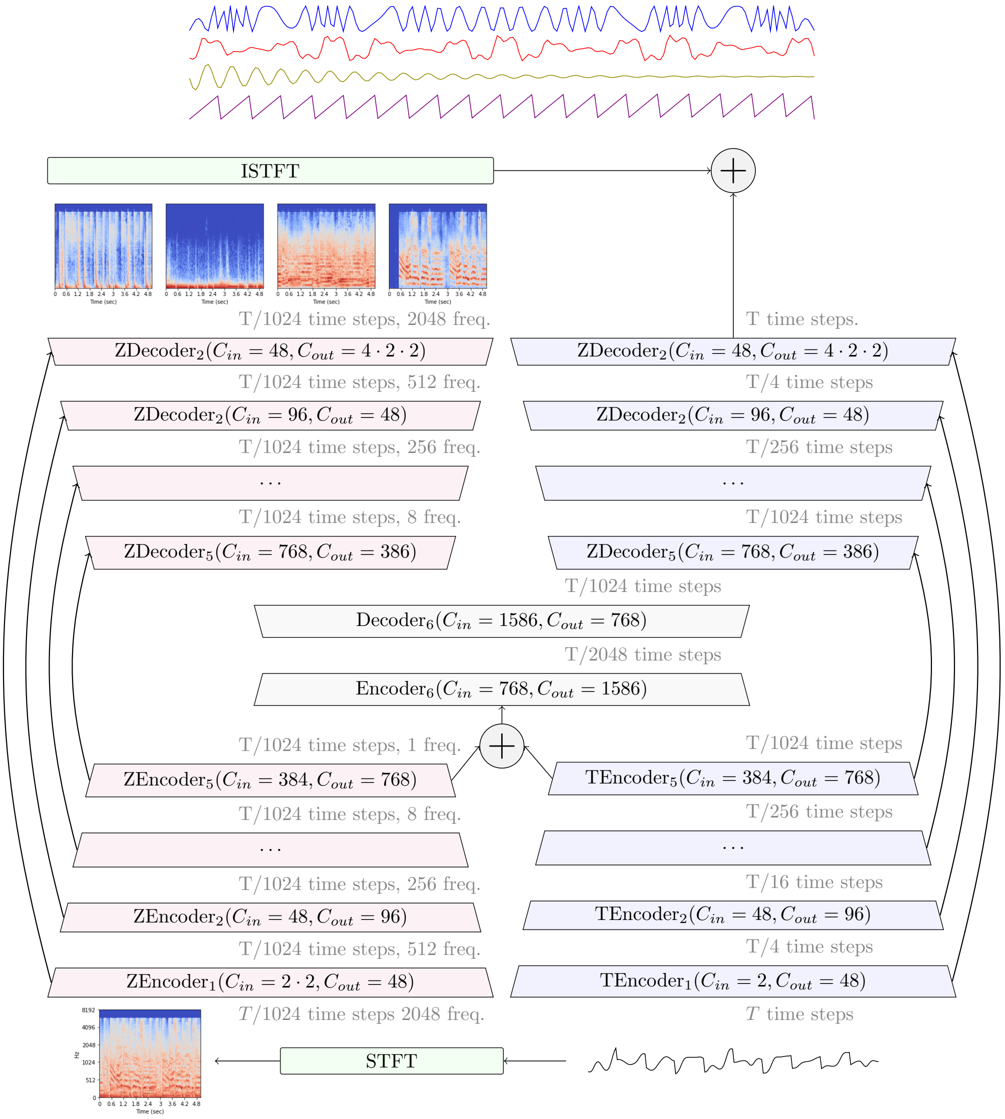
ISTFT
T/1024 time steps, 2048 freq.
T time steps.
ZDecoder$_2$($C_{in} = 48, C_{out} = 4 \cdot 2 \cdot 2$)
ZDecoder$_2$($C_{in} = 48, C_{out} = 4 \cdot 2 \cdot 2$)
T/1024 time steps, 512 freq.
T/4 time steps
ZDecoder$_2$($C_{in} = 96, C_{out} = 48$)
ZDecoder$_2$($C_{in} = 96, C_{out} = 48$)
T/1024 time steps, 256 freq.
T/256 time steps
...
...
T/1024 time steps, 8 freq.
T/1024 time steps
ZDecoder$_5$($C_{in} = 768, C_{out} = 386$)
ZDecoder$_5$($C_{in} = 768, C_{out} = 386$)
T/1024 time steps
Decoder$_6$($C_{in} = 1586, C_{out} = 768$)
T/2048 time steps
Encoder$_6$($C_{in} = 768, C_{out} = 1586$)
T/1024 time steps, 1 freq.
T/1024 time steps
ZEncoder$_5$($C_{in} = 384, C_{out} = 768$)
TEncoder$_5$($C_{in} = 384, C_{out} = 768$)
T/1024 time steps, 8 freq.
T/256 time steps
...
...
T/1024 time steps, 256 freq.
T/16 time steps
ZEncoder$_2$($C_{in} = 48, C_{out} = 96$)
TEncoder$_2$($C_{in} = 48, C_{out} = 96$)
T/1024 time steps, 512 freq.
T/4 time steps
ZEncoder$_1$($C_{in} = 2 \cdot 2, C_{out} = 48$)
TEncoder$_1$($C_{in} = 2, C_{out} = 48$)
T/1024 time steps 2048 freq.
T time steps
STFT
Time (sec)
Hz

Figure 1: Hybrid Demucs architecture. The input waveform is processed both through a temporal encoder, and first through the STFT followed by a spectral encoder. The two representations are summed when their dimensions align. The opposite happens in the decoder. The output spectrogram go through the ISTFT and are summed with the waveform outputs, giving the final model output. The Z prefix is used for spectral layers, and T prefix for the temporal ones.

Compressed residual branches

The original Demucs encoder layer is composed of a convolution with kernel size of 8 and stride of 4, followed by a ReLU, and of a convolution with kernel size of 1 followed by a GLU. Between those two convolutions, we introduce a novel compressed residual branch, composed of dilated convolutions, and for the innermost layers, of a biLSTM with limited span and local attention layer.

There are two compressed residual branch per encoder layer. Both are composed of a convolution with a kernel size of 3, stride of 1, dilation of 1 for the first branch and 2 for the second, and times less output dimensions than the input, followed by layer normalization (Ba et al., 2016) and a GELU activation.

For layer 4, 5 and 6 of the encoder (with the 6-th layer being shared by the spectral and temporal branch), long range context is processed through a local attention layer (see definition hereafter) as well as a biLSTM with 2 layers, inserted with a skip connection, and with a maximum span of 200 steps. In practice, the input is splitted into frames of 200 time steps, with a stride of 100 time steps. Each frame is processed concurrently by the biLSTM. Then, for any time step, the output from the frame for which it is the furthest away from the edge is kept.

Finally, and for all layers, a final convolution with a kernel size of 1 outputs twice as many channels as the input dimension of the residual branch, followed by a GLU. This output is then summed with the original input, after having been scaled through a LayerScale layer (Touvron et al., 2021), with an initial scale of $1\mathrm{e}{-3}$. A complete representation of the compressed residual branches is given on Figure 2.

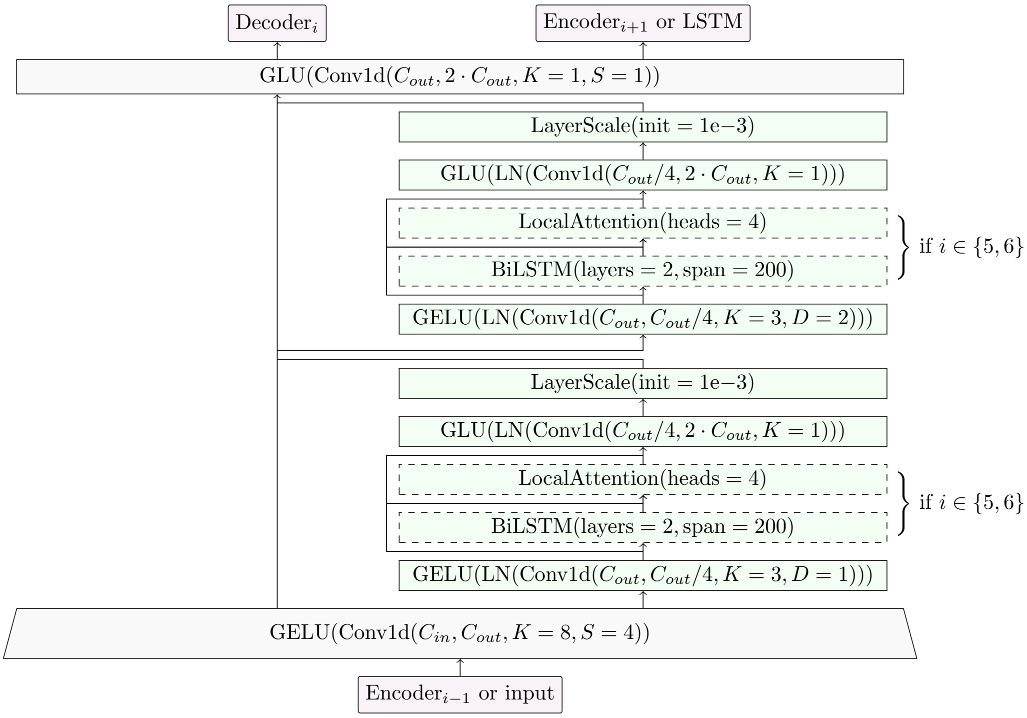
Decoder_i
Encoder_{i+1} or LSTM
$\text{GLU}(\text{Conv1d}(C_{out}, 2 \cdot C_{out}, K = 1, S = 1))$
$\text{LayerScale}(\text{init} = 1\text{e}{-3})$
$\text{GLU}(\text{LN}(\text{Conv1d}(C_{out}/4, 2 \cdot C_{out}, K = 1)))$
$\text{LocalAttention}(\text{heads} = 4)$
$\text{BiLSTM}(\text{layers} = 2, \text{span} = 200)$
if $i \in \{5, 6\}$
$\text{GELU}(\text{LN}(\text{Conv1d}(C_{out}, C_{out}/4, K = 3, D = 2)))$
$\text{LayerScale}(\text{init} = 1\text{e}{-3})$
$\text{GLU}(\text{LN}(\text{Conv1d}(C_{out}/4, 2 \cdot C_{out}, K = 1)))$
$\text{LocalAttention}(\text{heads} = 4)$
$\text{BiLSTM}(\text{layers} = 2, \text{span} = 200)$
if $i \in \{5, 6\}$
$\text{GELU}(\text{LN}(\text{Conv1d}(C_{out}, C_{out}/4, K = 3, D = 1)))$
$\text{GELU}(\text{Conv1d}(C_{in}, C_{out}, K = 8, S = 4))$
Encoder_{i-1} or input

Figure 2: Representation of the compressed residual branches that are added to each encoder layer. For the 5th and 6th layer, a BiLSTM and a local attention layer are added.

Local attention

Local attention builds on regular attention (Vaswani et al., 2017) but replaces positional embedding by a controllable penalty term that penalizes attending to positions that are far away. Formally, the attention weights from position i to position j is given by

$$w_{i,j} = \text{softmax}(Q_i^T K_j - \sum_{k=1}^{4} k\beta_{i,k}|i-j|)$$

where Q_i are the queries and K_j are the keys. The value $\beta_{i,k}$ is obtained as the output of a linear layer, initialized so that it is initially very close to 0. Having multiple $\beta_{i,k}$ with different weights k allows the network to efficiently reduce its receptive field without requiring $\beta_{i,k}$ to take large values. In practice, we use a sigmoid activation to derive the values $\beta_{i,k}$.

Stabilizing training

We observed that Demucs training could be unstable, especially as we added more layers and increased the training set size with 50 extra songs. Loading the model just before its divergence point, we realized that the weights for the innermost encoder and decoder layers would get very large eigen values.

A first solution is to use group normalization (with a 4 groups) just after the non residual convolutions for the layers 5 and 6 of the encoder and the decoder. Using normalization on all layers will deteriorate performance, but using it only on the innermost layer seems to stabilize training without hurting performance. Interestingly, when the training is stable (in particular when trained only on MusDB), using normalization was at best neutral with respect to the separation score, but never improved it, and considerably slowed down training during the first half of the epochs. When the training was unstable, using normalization would improve the overall performance as it allows the model to train for a larger number of epochs.

A second solution we investigated was to use singular value regularization (Yoshida & Miyato, 2017). While previous work used the power method iterative procedure, we obtained better and faster approximations of the largest singular value using a low rank SVD method (Halko et al., 2011). This solution had the advantage of always improving generalization, even when the training was already stable. Sadly, it was not sufficient on its own to remove entirely instabilities, but only to reduce them. Another down side was the longer training time due to the extra low rank SVD evaluation.

In the end, in order to both achieve the best performance and remove entirely training instabilities, the two solutions were combined.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv Preprint arXiv:1607.06450*.

Choi, W., Kim, M., Chung, J., & Jung, S. (2021). LaSAFT: Latent source attentive frequency transformation for conditioned source separation. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 171–175.

Choi, W., Kim, M., Chung, J., Lee, D., & Jung, S. (2020). Investigating u-nets with various intermediate blocks for spectrogram-based singing voice separation. In ISMIR (Ed.), *21th international society for music information retrieval conference*.

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). Language modeling with gated convolutional networks. *Proceedings of the International Conference on Machine Learning*.

Défossez, A., Usunier, N., Bottou, L., & Bach, F. (2019). Music source separation in the waveform domain. *arXiv Preprint arXiv:1911.13254*.

Halko, N., Martinsson, P.-G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, *53*(2), 217–288.

Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv Preprint arXiv:1606.08415*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Isik, U., Giri, R., Phansalkar, N., Valin, J.-M., Helwani, K., & Krishnaswamy, A. (2020). Poconet: Better speech enhancement with frequency-positional embeddings, semi-supervised conversational data, and biased loss. *arXiv Preprint arXiv:2008.04470*.

Jansson, A., Humphrey, E., Montecchio, N., Bittner, R., Kumar, A., & Weyde, T. (2017). Singing voice separation with deep u-net convolutional networks. *ISMIR 2018*.

Kong, Q., Cao, Y., Liu, H., Choi, K., & Wang, Y. (2021). Decoupling magnitude and phase estimation with deep ResUNet for music source separation. *22th International Society for Music Information Retrieval Conference*.

Lluís, F., Pons, J., & Serra, X. (2018). End-to-end music source separation: Is it possible in the waveform domain? *arXiv Preprint arXiv:1810.12*.

Luo, Y., Chen, Z., & Yoshioka, T. (2020). Dual-path rnn: Efficient long sequence modeling for time-domain single-channel speech separation. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 46–50.

Luo, Y., & Mesgarani, N. (2019). Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Mitsufuji, Y., Fabbro, G., Uhlich, S., & Stöter, F.-R. (2021). Music demixing challenge 2021. *arXiv Preprint arXiv:2108.13559*.

Nugraha, A. A., Liutkus, A., & Vincent, E. (2016). Multichannel music separation with deep neural networks. *Signal Processing Conference (EUSIPCO), 2016 24th European*.

Pons, J., Pascual, S., Cengarle, G., & Serrà, J. (2021). Upsampling artifacts in neural audio synthesis. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3005–3009.

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., & Bittner, R. (2017). *The MUSDB18 corpus for music separation*.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*.

Sawata, R., Uhlich, S., Takahashi, S., & Mitsufuji, Y. (2020). *All for one and one for all: Improving music separation by bridging networks*. http://arxiv.org/abs/2010.04228

Stoller, D., Ewert, S., & Dixon, S. (2018). Wave-u-net: A multi-scale neural network for end-to-end audio source separation. *arXiv Preprint arXiv:1806.03185*.

Stöter, F.-R., Liutkus, A., & Ito, N. (2018). The 2018 signal separation evaluation campaign. *14th International Conference on Latent Variable Analysis and Signal Separation*.

Stöter, F.-R., Uhlich, S., Liutkus, A., & Mitsufuji, Y. (2019). Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software*. https://doi.org/10.21105/joss.01667

Takahashi, N., & Mitsufuji, Y. (2020). D3Net: Densely connected multidilated DenseNet for music source separation. *arXiv Preprint arXiv:2010.01733*.

Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., & Jégou, H. (2021). Going deeper with image transformers. *arXiv Preprint arXiv:2103.17239*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 5998–6008.

Yoshida, Y., & Miyato, T. (2017). Spectral norm regularization for improving the generalizability of deep learning. *arXiv Preprint arXiv:1705.10941*.

Yu, F., & Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. *ICLR*.