Demucs: Deep Extractor for Music Sources with extra unlabeled data remixed

Alexandre Défossez, Nicolas Usunier, Léon Bottou, Francis Bach

Demucs: Deep Extractor for Music Sources with extra unlabeled data remixed

Alexandre Défossez
Facebook AI Research
INRIA / École Normale Supérieure
PSL Research University
Paris, France
defossez@fb.com

Nicolas Usunier
Facebook AI Research
Paris, France
usunier@fb.com

Léon Bottou
Facebook AI Research
New York, USA
leonb@fb.com

Francis Bach
INRIA / École Normale Supérieure
PSL Research University
Paris, France
francis.bach@ens.fr

Abstract

We study the problem of source separation for music using deep learning with four known sources: drums, bass, vocals and other accompaniments. State-of-the-art approaches predict soft masks over mixture spectrograms while methods working on the waveform are lagging behind as measured on the standard MusDB [22] benchmark. Our contribution is two fold. (i) We introduce a simple convolutional and recurrent model that outperforms the state-of-the-art model on waveforms, that is, Wave-U-Net [28], by 1.6 points of SDR (signal to distortion ratio). (ii) We propose a new scheme to leverage unlabeled music. We train a first model to extract parts with at least one source silent in unlabeled tracks, for instance without bass. We remix this extract with a bass line taken from the supervised dataset to form a new weakly supervised training example. Combining our architecture and scheme, we show that waveform methods can play in the same ballpark as spectrogram ones.

1 Introduction

Cherry first noticed the "cocktail party effect" [5]: how the human brain is able to separate a single conversation out of a surrounding noise from a room full of people chatting. Bregman later tried to understand how the brain was able to analyse a complex auditory signal and segment it into higher level streams. His framework for auditory scene analysis [4] spawned its computational counterpart, trying to reproduce or model accomplishment of the brains with algorithmic means [36].

When producing music, recordings of individual instruments called *stems* are arranged together and mastered into the final song. The goal of source separation is then to recover those individual stems from the mixed signal. Unlike the cocktail problem, there is not a single source of interest to differentiate from an unrelated background noise, but instead a wide variety of tones and timbres playing in a coordinated way. As part of the SiSec Mus evaluation campaign for music separation [29], a choice was made to regroup those individual stems into 4 broad categories: (1) `drums`, (2) `bass`, (3) `other`, (4) `vocals`.

Each source is represented by a waveform $s_i \in \mathbb{R}^{C,T}$ where C is the number of channels (1 for mono, 2 for stereo) and T the number of samples. We define $\mathbf{s} := (s_i)_i$ the concatenation of sources in a

Preprint version.

tensor of size $(4, C, T)$ and the mixture $s := \sum_{i=1}^{4} s_i$. We aim at training a model that minimises

$$\min_{\theta} \sum_{\mathbf{s} \in \mathcal{D}} \sum_{i=1}^{4} l(g_\theta^i(s), s_i) \tag{1}$$

for some dataset $\mathcal{D}$, reconstruction error l, model architecture g with 4 outputs g^i, and model weights $\theta \in \mathbb{R}^d$.

As presented in the next section, most methods to solve (1) learn a mask per source σ_i on the mixture spectrogram $S := \mathrm{STFT}(s)$ (Short-Time Fourier Transform). The estimated sources are then $\hat{s}_i := \mathrm{ISTFT}(\sigma_i S)$ (Inverse Short-Time Fourier Transform). The mask σ_i can either be a binary mask valued in $\{0, 1\}$ or a soft assignment valued in $[0, 1]$. Those methods are state-of-the-art and perform very well without requiring large models. However, they come with two limitations:

1. There is no reason for $\sigma_i S$ to be a real spectrogram (i.e., obtained from a real signal). In that case the ISTFT step will perform a projection step that is not accounted for in the training loss and could result in artifacts.
2. Such methods do not try to model the phase but reuse the input mixture. Let us imagine that a guitar plays with a singer at the same pitch but the singer is doing a slight vibrato, i.e., a small modulation of the pitch. This modulation will impact the spectrogram phase, as the derivative of the phase is the instant frequency. Let say both the singer and the guitar have the same intensity, then the ideal mask would be 0.5 for each. However, as we reuse for each source the original phase, the vibrato from the singer would also be applied to the guitar. While this could be consider a corner case, its existence is a motivation for the search of an alternative.

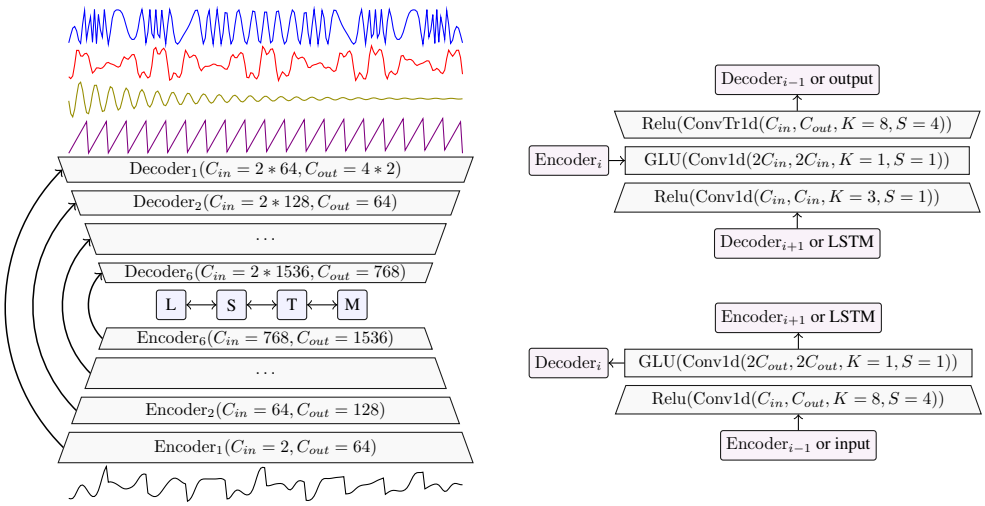
Learning a model from/to the waveform could allow to lift some of the aforementioned limitations. Because a waveform is directly generated, the training loss is end-to-end, with no extra synthesis step that could add artifacts which solves the first point above. As for the second point, it is unknown whether any model could succeed in separating such a pathological case. In the fields of speech or music generation, direct waveform synthesis has replaced spectrogram based methods [34, 20, 7]. When doing generation without an input signal s, the first point is more problematic. Indeed, there is no input phase to reuse and the inversion of a power spectrogram will introduce significant artifacts [8]. Those successes were also made possible by the development of very large scale datasets (30GB for the NSynth dataset [8]). In comparison the standard MusDB dataset is only a few GB. This explains, at least partially, the worse performance of waveform methods for source separation [29].

In this paper we aim at taking waveform based methods one step closer to spectrogram methods. We contribute a simple model architecture inspired by previous work in source separation from the waveform and audio synthesis. We show that this model outperforms the previous state of the art on the waveform domain. Given the limited data available, we further refine the performance of our model by using a novel semi-supervised data augmentation scheme that allows to leverage 2,000 unlabeled songs.

2 Related Work

A first category of methods for supervised music source separation work on power spectrograms. They predict a power spectrogram for each source and reuse the phase from the input mixture to synthesise individual waveforms. Traditional methods have mostly focused on blind (unsupervised) source separation. Non-negative matrix factorization techniques [26] model the power spectrum as a weighted sum of a learnt spectral dictionary, whose elements can then be grouped into individual sources. Independent component analysis [12] relies on independence assumptions and multiple microphones to separate the sources. Learning a soft/binary mask over power spectrograms has been done using either HMM-based prediction [25] or segmentation techniques [3].

With the development of deep learning, fully supervised methods have gained momentum. Initial work was performed on speech source separation [9] then for music using simple fully connected networks over few spectrogram frames [32], LSTMs [33], or multi scale convolutional / recurrent networks [18, 30, 31]. State-of-the-art performance is obtained with those models when trained with extra labeled data. We show that our model architecture combined with our semi-supervised scheme can provide performance almost on par, while being trained on 5 times less labeled data.



(a) Demucs architecture with the mixture waveform as input and the four sources estimates as output. Arrows represents U-Net connections.

(b) Detailed view of the layers Decoder_i on the top and Encoder_i on the bottom. Arrows represent connections to other parts of the model.

Figure 1: Demucs complete architecture on the left, with detailed representation of the encoder and decoder layers on the right. Key novelties compared to the previous Wave-U-Net are the GLU activation in the encoder and decoder, the bidirectional LSTM in-between and exponentially growing number of channels, allowed by the stride of 4 in all convolutions.

On the other hand, working directly on the waveform only became possible with deep learning models. A Wavenet-like but regression based approach was first used for speech denoising [23] and then adapted to source separation [19]. Concurrently, a convolutional network with a U-Net structure called Wave-U-Net was used first on spectrograms [14] and then adapted to the waveform domain [28]. Those methods performs significantly worse than the spectrogram ones as shown in the latest SiSec Mus source separation evaluation campaign [29]. As shown in Section 5, we outperform Wave-U-Net by a large margin with our architecture alone.

In [21], the problem of semi-supervised source separation is tackled for 2 sources separation where a dataset of mixtures and unaligned isolated examples of source 1 but not source 2 is available. Using specifically crafted adversarial losses the authors manage to learn a separation model. In [11], the case of blind, i.e., completely unsupervised source separation is covered, combining adversarial losses with a remixing trick similar in spirit to our unlabeled data remixing presented in Section 4. Both papers are different from our own setup, as they assume that they completely lack isolated audio for some or all sources. Finally, when having extra isolated sources, previous work showed that it was possible to use adversarial losses to leverage them without using them to generate new mixtures [27]. Unfortunately, extra isolated sources are exactly the kind of data that is hard to come by. As far as we know, no previous work tried to leverage unlabeled songs in order to improve supervised source separation performance. Besides, most previous work relied on adversarial losses, which can prove expensive while our remixing trick allows for direct supervision of the training loss.

3 Model Architecture

Our network architecture is a blend of ideas from the SING architecture [7] developed for music note synthesis and Wave-U-Net. We reuse the synthesis with large strides and large number of channels as well as the combination of a LSTM and convolutional layers from SING, while retaining the U-Net [24] structure of Wave-U-Net. The model is composed of a convolutional encoder, an LSTM and a convolutional decoder, with the encoder and decoder linked with skip U-Net connections. The model takes a stereo mixture $s = \sum_i s_i$ as input and outputs a stereo estimate $\hat{s}_i$ for each source. Similarly to other work in generation in both image [16, 15] and sound [7], we do not use batch normalization [13] as our early experiments showed that it was detrimental to the model performance.

The encoder is composed of $L := 6$ stacked layers numbered from 1 to L. Layer i is composed of a convolution with kernel size $K := 8$, stride $S := 4$, C_{i-1} input channels, C_i output channels and ReLU activation followed by a 1x1 convolution with GLU activation [6]. As the GLU outputs $C/2$ channels with C channels as input, we double the number of channels in the 1x1 convolution. We define $C_0 := 2$ the number of channels in the input mixture and $C_1 := 48$ the initial number of channels for our model. For $i \in \{2, \dots, L\}$ we take $C_i := 2C_{i-1}$ so that the final number of channels is $C_L = 1536$. We then use a bidirectional LSTM with 2 layers and a hidden size C_L. The LSTM outputs $2C_L$ channels per time position. We use a 1x1 convolution with ReLU activation to take that number down to C_L.

The decoder is almost the symmetric of the encoder. It is composed of L layers numbered in reverse order from L to 1. The i-th layer starts with a convolution with kernel size 3 and stride 1, input/output channels C_i and a ReLU activation. We concatenate its result with the output of the i-th layer of the encoder to form a U-Net and take back the number of channels to C_i using a 1x1 convolution with GLU activation. Finally, we use a transposed convolution with kernel size $K = 8$ and stride $S = 4$, C_{i-1} outputs and ReLU activation. For the final layer, we instead output $4C_0$ channels and do not use any activation function.

Weights rescaling The weights of a convolutional layer in a deep learning model are usually initialized in a way that account for the number of input channels and receptive field of the convolutions (i.e., fan in), as introduced by He et al. [10]. The initial weights of a convolution will roughly scale as $\frac{1}{\sqrt{KC_{in}}}$ where K is the kernel size and C_{in} the number of input channels. For instance, the standard deviation after initialization of the weights of the first layer of our encoder is about 0.2, while that of the last layer is 0.01. Modern optimizers such as Adam [17] normalize the gradient update per coordinate so that, on average, each weight will receive updates of the same magnitude. Thus, if we want to take a learning rate large enough to tune the weights of the first layer, it will most likely be too large for the last layer.

In order to remedy this problem, we use a trick that is equivalent to using specific learning rates per layer. Let us denote w the weights at initialization used to compute the convolution $w * x$. We take $\alpha := \mathrm{std}(w)/a$, where a is a reference scale. We replace w by $w' = w/\sqrt{\alpha}$ and the output of the convolution by $\sqrt{\alpha}w' * x$, so that the output of the layer is unchanged. This is similar to the equalized learning rate trick used for image generation with GAN [16]. We observed both faster decay of the training loss and convergence to a better optimum when using the weight rescaling trick, see Section 5.3. Optimal performance was obtained for a reference level $a := 0.1$. We also tried rescaling the weights by $1/\alpha$ rather than $1/\sqrt{\alpha}$ however this made the training loss diverge.

Synthesis vs. filtering Let us denote $e_i(s)$ the output of the i-th layer of the encoder and $d_i(s)$ the output of the i-th layer of the decoder. Wave-U-Net takes $d_i(s)$ and upsamples it using linear interpolation. It then concatenates it with $e_{i-1}(s)$ (with $e_0(s) := s$) and applies a convolution with a stride of 1 to obtain $d_{i-1}(s)$. Thus, it works by taking a coarse representation, upsampling it, adding back the fine representation from $e_{i-1}(s)$ and filtering it out to separate channels.

On the other hand, our model takes $d_i(s)$ and concatenates it with $e_i(s)$ and uses a transposed convolution to obtain $d_{i-1}(s)$. A transposed convolution is different from a linear interpolation upsampling. With a sufficient number of input channels, it can generate any signal, while a linear upsampling will generate a signal with higher sampling rate but no high frequencies. High frequencies are injected using a U-Net skip connection. Separation is performed by applying various filters to the obtained signal (aka convolution with a stride of 1).

Thus, Wave-U-Net generates its output by iteratively upsampling, adding back the high frequency part of the signal from the matching encoder output (or from the input for the last decoder layer) and filtering. On the other hand, our approach consist in a direct synthesis. The main benefit of synthesis is that we can use a relatively large stride in the decoder, thus speeding up the computations and allowing for a larger number of channels. We believe this larger number of channels is one of the reasons for the better performance of our model as shown in Section 5.

4 Unlabeled Data Remixing

In order to leverage unlabeled songs, we propose to first train a classifier to detect the absence or presence of each source on small time frames, using a supervised train set for which we know the

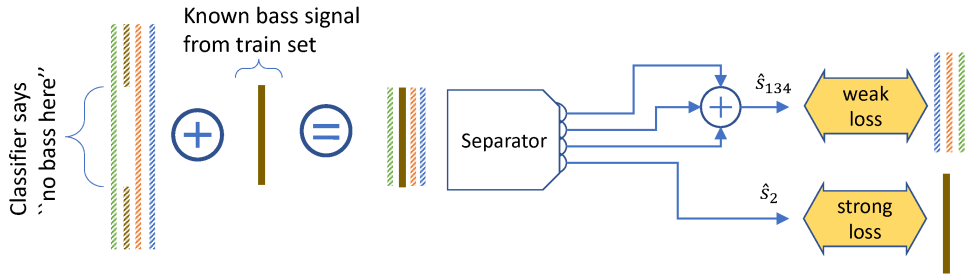


Figure 2: Overall representation of our unlabeled data remixing pipeline. When we detect an excerpt of at least 5 seconds with one source silent, here the `bass`, we recombine it with a single bass sample from the training set. We can then provide strong supervision for the silent source, and weak supervision for the other 3 as we only know the ground truth for their sum.

contribution of each source. When we detect an audio excerpt m_i with at least 5 seconds of silence for source i, we add it to a new set $\mathcal{D}_i$. We can then mix an example $m_i \in \mathcal{D}_i$ with a single source s_i taken from the supervised train set in order to form a new mixture $s' = s_i + m_i$, noting s'_j the ground truth for this example (potentially unknown to us) for each source j. As the source i is silent in m_i we can provide strong supervision for source i as we have $s'_i = s_i$ and weak supervision for the other sources as we have $\sum_{j \neq i} s'_j = m_i$. The whole process pipeline is represented in Figure 2.

Motivation for this approach comes from our early experiments with the available supervised data which showed a clear tendency for overfitting when training our separation models. We first tried using completely unsupervised regularization, for instance given an unlabeled track m, we want $\sum_i \hat{s}_i = m$ where $\hat{s}_i$ is the estimated source i. This proved too weak to improve performance. We then tried to detect passages with a single source present however this proved too rare of an event in Pop/Rock music: for the standard MusDB dataset presented in Section 5.1, source `other` is alone 2.6% of the time while the others are so less than 0.5% of the time. Accounting for the fact that our model will never reach a recall of 100%, this represents too few extractable data to be interesting. On the other hand, a source being silent happen quite often, 11% of the time for the `drums`, 13% for the `bass` or 32% for the `vocals`. This time, the `other` is the least frequent with 2.6% and hardest to extract as noted hereafter.

We first formalize our classification problem and then describe the extraction procedure. The use of the extracted data for training is detailed in Section 5.2.

4.1 Silent source detector

Given a mixture $s = \sum_i s_i$, we define for all sources i the relative volume $V_i(\mathbf{s}) := 10 \log_{10} \frac{\|s_i\|^2}{\|s\|^2}$ and a source being silent as $S_i(\mathbf{s}) := \mathbb{1}_{V_i(\mathbf{s}) \geq V_{\text{thres}}}$. For instance, having $V_i = -20$ means that source i is 100 times quieter than the mixture. Doing informal testing, we observe that a source with a relative volume between 0 and -10 will be perceived clearly, between -10 and -20 it will feel like a whisper and almost silent between -20 and -30. A source with a relative volume under -30 is perceptually zero.

We can then train a classifier to estimate $P_i := \mathbb{P}\{S_i = 1 | s\}$, the probability that source i is silent given the input mixture s. Given the limited amount of supervised data, we use a Wavelet scattering transform [1] of order two as input features rather than the raw waveform. This transformation is computed using the Kymatio package [2]. The model is then composed of convolutional layers with max pooling and batch normalization and a final LSTM that produces an estimate $\hat{P}_i$ for every window of 0.64 seconds with a stride of 64 ms. We detail the architecture in the Section 2 of the supplementary material. We have observed that using a downsampled (16kHz) and mono representation of the mixture further helps prevent overfitting.

The silence threshold is set to -13dB. Although this is far from silent, this allows for a larger number of positive samples and better training. We empirically observe that the true relative volumes decreases as the estimated probability $\hat{P}_i$ increases. Thus, in order to select only truly silent samples ($V_i \leq -30$), one only needs to select a high threshold on $\hat{P}_i$.

Table 1: Comparison of Demucs with the state of the art in the waveform domain (Wave-U-Net) and in the spectrogram domain (MMDenseNet, MMDenseNetLSTM) on the MusDB test set. Extra data is the number of extra songs used, either labeled with the waveform for each source or unlabeled. We report the median over all tracks of the median SDR over each track, as done in the SiSec Mus evaluation campaign [29]. For easier comparison, the `All` column is obtained by concatenating the metrics from all sources and then taking the median.

		Extra data		Test SDR in dB				
Architecture	**Wav?**	*labeled*	*unlabed*	`All`	`Drums`	`Bass`	`Other`	`Vocals`
MMDenseNet	✗	✗	✗	5.34	6.40	5.14	4.13	6.57
Wave-U-Net	✓	✗	✗	3.17	4.16	3.17	2.24	3.05
Demucs	✓	✗	✗	4.81	5.38	5.07	3.01	5.44
Demucs	✓	✗	2,000	5.09	5.79	6.23	3.45	5.51
Demucs	✓	100	✗	5.41	5.99	5.72	3.65	6.17
Demucs	✓	100	2,000	5.67	6.50	6.21	3.80	6.21
MMDenseLSTM	✗	804	✗	5.97	6.75	5.28	4.72	7.15
MMDenseNet	✗	804	✗	5.85	6.81	5.21	4.37	6.74

Table 2: Ablation study for the novel elements in our architecture or training procedure. Unlike on Table 1, we report the simple SDR defined in Section 5.1 rather than the extended version $\mathrm{SDR}_{\mathrm{ext}}$. We also report average values rather than medians as this make small changes more visible. This explains the SDR reported here being much smaller than on Table 1. Both metrics are averaged over the last 3 epochs and computed on the MusDB test set. Reference is trained with remixed unlabeled data, with stereo channels input resampled at 22kHz, on the train set of MusDB.

	Train set		**Test set**	
Difference	L1 loss	SDR	L1 loss	SDR
Reference	0.090	8.82	0.169	5.09
no remixed	0.089	8.87	0.175	4.81
no GLU	0.099	8.00	0.174	4.68
no BiLSTM	0.156	8.42	0.182	4.83
MSE loss	N/A	8.84	N/A	5.04
no weight rescaling	0.094	8.39	0.171	4.68

4.2 Extraction procedure

We assume we have a few labeled data from the same distribution as the unlabeled data available, in our case we used 100 labeled tracks for 2,000 unlabeled ones, as explained in Section 5.1. If such data is not available, it is still possible to annotate part of the unlabeled data, but only with weak labels (source present or absence) which is easier than obtaining the exact waveform for each source. We perform extraction by first setting thresholds probabilities p_i for each source. We define p_i as the lowest limit so that for at least 95% of the samples with $\hat{P}_i(s) \geq p_i$, we have $V_i(\mathbf{s}) \leq -20$ on the stem set. We then only keep audio extracts where $\hat{P}_i \geq p_i$ for at least 5 seconds, which reduces the amount of data extracted by roughly 50% but also reduces the 95% percentile of the relative volume from -20 to -30. We assemble all the 5 seconds excerpt where source i is silent into a new dataset $\mathcal{D}_i$.

In our case, we did not manage to obtain more than a few minutes of audio with source `other` silent. Indeed, as noted above, it is the most frequent source, training examples without it are rare leading to unreliable prediction.

5 Experimental results

We present here the datasets, metrics and baselines used for evaluating our architecture and unlabeled data remixing. We mostly reuse the framework setup for the SiSec Mus evaluation campaign for music source separation [29] and their MusDB dataset [22].

5.1 Evaluation framework

MusDB and unsupervised datasets We use the MusDB [22] dataset, which is composed of 150 songs with full supervision in stereo and sampled at 44100Hz. For each song, we have the exact waveform of the `drums`, `bass`, `other` and `vocals` parts, i.e. each of the sources. The actual song, a.k.a. the mixture, is the sum of those four parts. The first 100 songs form the *train set* while the remaining 50 are kept for the *test set*.

To test out the semi-supervised scheme described in Section 4, we exploit our own set of 2,000 unlabeled tracks, which represents roughly 4.5 days of audio. It is composed of 4% of Heavy Metal, 4% of Jazz, 37% of Pop and 55% of Rock music. Although we do not release this set, we believe that a similarly composed digital music collection will allow to replicate our results. We refer to this data as the *unsupervised* or *unlabled* set.

We also collected raw stems for 100 tracks, i.e., individual instrument recordings used in music production software to make a song. Those tracks come from the same distribution as our unsupervised dataset but do not overlap. We manually assigned each instrument to one of the sources using simple rules on the filenames (for instance "Lead Guitar.wav" is assigned to the `other` source) or listening to the stems in case of ambiguity. We will call this extra supervised data the *stem set*. As some of the baslises used additional labeled data (807 songs), we also provide metrics for our own architecture trained using this extra stem set.

We applied our extraction pipeline to the 2,000 unlabeled songs, and obtained about 1.5 days of audio (with potential overlap due to our extraction procedure) for with the source `drums`, `bass` or `vocals` silent which form respectively the datasets $\mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_3$. We could not retrieve a significant amount of audio for the `other` source. Indeed, this last source is the most frequently present (there is almost always a melodic part in a song), and with the amount of data available, we could not train a model that would reliably predict the absence of this source. As a consequence, we do not extract a dataset $\mathcal{D}_2$ for it. We did manage to extract a few hours with only the `other` source, but we have not tried to inject it into our separation model training. Although we trained our model on mono and 16kHz audio, we perform the extraction on the original 44kHz stereo data.

Source separation metrics Measurements of the performance of source separation models was developed by Vincent et al. for blind source separation [35] and reused for supervised source separation in the SiSec Mus evaluation campaign [29]. Reusing the notations from [35], let us take a source $j \in 1, 2, 3, 4$ and introduce P_{s_j} (resp $P_{\mathbf{s}}$) the orthogonal projection on s_j (resp on $\mathrm{Span}(s_1, \ldots, s_4)$). We then take with $\hat{s}_j$ the estimate of source s_j, $s_{\text{target}} := P_{s_j}(\hat{s}_j)$, $e_{\text{interf}} := P_{\mathbf{s}}(\hat{s}_j) - P_{s_j}(\hat{s}_j)$ and $e_{\text{artif}} := \hat{s}_j - P_{\mathbf{s}}(\hat{s}_j)$. The signal to distortion ratio is then defined as

$$\mathrm{SDR} := 10 \log_{10} \frac{\|s_{\text{target}}\|^2}{\|e_{\text{interf}} + e_{\text{artif}}\|^2}. \tag{2}$$

Note that this definition is invariant to the scaling of $\hat{s}_j$. We used the python package `museval`[1] which provide a reference implementation for the SiSec Mus 2018 evaluation campaign. It also allows time invariant filters to be applied to $\hat{s}_j$ as well as small delays between the estimate and ground truth [35]. As done in the SiSec Mus competition, we report the median over all tracks of the median of the metric over each track computed using the `museval` package. Similarly to previous work [28, 30, 31], we focus in this section on the SDR, but other metrics can be defined (SIR an SAR) and we present them in the Appendix, Section B.

Baselines We selected the best performing models from the last SiSec Mus evaluation campaign [29] as baselines. MMDenseNet [30] is a multiscale convolutional network with skip and U-Net connections. This model was submitted as TAK1 when trained without extra labeled data and as TAK3 when trained with 804 extra labeled songs[2]. MMDenseLSTM [31] is an extension of MMDenseNet that adds LSTMs at different scales of the encoder and decoder. This model was submitted as TAK2 and was trained with the same 804 extra labeled songs. Unlike MMDenseNet, this model was not submitted without supplementary training data. The only Waveform based method submitted to the evaluation campaign is Wave-U-Net [28] with the identifier STL2. Metrics from all baselines were downloaded from the SiSec submission repository[3].

[1] `https://github.com/sigsep/sigsep-mus-eval`

[2] Source: `https://sisec18.unmix.app/#/methods/TAK2`

[3] `https://github.com/sigsep/sigsep-mus-2018`

5.2 Training procedure

We define one epoch over the dataset as a pass over all 5 second extracts with a stride of 0.5 seconds. We train the classifier described in Section 4 on 4 V100 GPUs for 40 epochs with a batch size of 64 using Adam [17] with a learning rate of 5e-4. We use the sum of the binary cross entropy loss for each source as a training loss. The Demucs separation model described in Section 3 is trained for 400 epochs on 16 V100 GPUs, with a batch size of 128 using Adam with a learning rate of 5e-4 and decaying the learning rate every 160 epochs by a factor of 5. We perform the following data augmentation, partially inspired by [33]: shuffling sources within one batch, randomly shifting sources in time (same shift for both channels), randomly swapping channels, random multiplication by ± 1 per channel. Given the cost of fitting those models, we perform a single run for each configuration.

We use the L1 distance between the estimated sources $\hat{s}_i$ and the ground truth s_i as we observed it improved the performance quite a bit, as shown on Table 2. We have tried replacing or adding to this loss the L1 distance between the power spectrogram of $\hat{s}_i$ and that of s_i, as done in [7], however it only degraded the final SDR of the model. When using the unlabeled data remixing trick describe in Section 4, we perform an extra step with probability 0.25 after each training batch from the main training step. We sample one source i at random out of (0) `drums`, (1) `bass` or (3) `vocals` (remember that we could not extract excerpt for source `other`) and obtain $m_i \in \mathcal{D}_\rangle$ where source i is silent and s_i from the training set where only i is present. We take $s' := m_i + s_i$ and perform a gradient step on the following loss

$$\|\hat{s}'_i - s_i\|_1 + \lambda \Big\| \sum_{j \neq i} \hat{s}'_j - m_i \Big\|_1. \tag{3}$$

Given that the extracted examples m_i are noisier than those coming from the train set, we use a separate instance of Adam for this step with a learning rate 10 times smaller than the main one. Furthermore, as we only have weak supervision over sources $j \neq i$, the second term is too be understood as a regularizer rather than a leading term, thus we take $\lambda := 10^{-6}$.

5.3 Evaluation results

We compare the performance of our approach with the state of the art in Table 1. On the top half, we show all methods trained without supplementary data. We can see a clear improvement coming from our new architecture alone compared to Wave-U-Net while MMDenseNet keeps a clear advantage. We then look at the impact of adding unlabeled remixed data. We obtain a gain of nearly 0.3 of the median SDR. As a reference, adding 100 labeled tracks to the train set gives a gain of 0.6. Interestingly, even when training with the extra tracks, our model still benefits from the unlabeled data, gaining an extra 0.2 points of SDR. MMDenseLSTM and MMDenseNet still obtain the best performance overall but we can notice that Demucs trained with unlabeled data achieved state of the art performance for the separation of the `bass` source. It only had access to 100 extra labeled songs, which is far from the 804 extra labeled songs used for MMDenseNet/LSTM and it would be interesting to see how waveform based models perform with a dataset that large. Box plots with quantiles can be found in the Appendix, Section B. Audio samples from different Demucs variant and the baselines are provided online[4]. We provide an ablation study of the main novelties of this paper on Table 2. on the train set of MusDB plus our remixed unlabeled data.

Conclusion

We presented Demucs, a simple architecture inspired by previous work in source separation from the waveform and audio synthesis that reduces the gap between spectrogram and waveform based methods from 2.2 points of median SDR to 0.5 points when trained only on the standard MusDB dataset. We have also demonstrated how to leverage 2,000 unlabeled mp3s by first training a classifier to detect excerpt with at least one source silent and then remixing it with an isolated source from the training set. To our knowledge, this is the first semi-supervised approach to source separation that does not rely on adversarial losses. Finally, training our model with remixed unlabeled data as well as 100 extra training examples, we obtain performance almost on par with that of state of the art spectrogram based methods, even better for the `bass` source, making waveform based method a legitimate contender for supervised source separation.

[4] `https://ai.honu.io/papers/demucs/`

References

[1] Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 2014.

[2] Mathieu Andreux, Tomás Angles, Georgios Exarchakis, Roberto Leonarduzzi, Gaspar Rochette, Louis Thiry, John Zarka, Stéphane Mallat, Eugene Belilovsky, and Joan Bruna. Kymatio: Scattering transforms in python. Technical Report 1812.11214, arXiv, 2018.

[3] Francis Bach and Michael I. Jordan. Blind one-microphone speech separation: A spectral learning approach. In *Advances in neural information processing systems*, 2005.

[4] A. S. Bregman. *Auditory Scene Analysis*. MIT Press, Cambridge, MA, 1990.

[5] E. Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the Acoustic Society of America*, 1953.

[6] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the International Conference on Machine Learning*, 2017.

[7] Alexandre Défossez, Neil Zeghidour, Usunier Nicolas, Leon Bottou, and Francis Bach. Sing: Symbol-to-instrument neural generator. In *Advances in Neural Information Processing Systems 32*, 2018.

[8] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. Technical Report 1704.01279, arXiv, 2017.

[9] Emad M. Grais, Mehmet Umut Sen, and Hakan Erdogan. Deep neural networks for single channel source separation. In *International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 2015.

[11] Yedid Hoshen. Towards unsupervised single-channel blind source separation using adversarial pair unmix-and-remix. Technical Report 1812.07504, arXiv, 2018.

[12] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*. John Wiley & Sons, 2004.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. Technical Report 1502.03167, arXiv, 2015.

[14] Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep u-net convolutional networks. In *ISMIR 2018*, 2017.

[15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. Technical Report 1710.10196, arXiv, 2017.

[16] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. Technical Report 1812.04948, arXiv, 2018.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[18] Jen-Yu Liu and Yi-Hsuan Yang. Denoising auto-encoder with recurrent skip connections and residual regression for music source separation. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018.

[19] Francesc Lluís, Jordi Pons, and Xavier Serra. End-to-end music source separation: is it possible in the waveform domain? Technical Report 1810.12187, arXiv, 2018.

[20] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. Technical Report 1612.07837, arXiv, 2016.

[21] Michael Michelashvili, Sagie Benaim, and Lior Wolf. Semi-supervised monaural singing voice separationwith a masking network trained on synthetic mixtures. Technical Report 1812.06087, arXiv, 2018.

[22] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The musdb18 corpus for music separation, 2017.

[23] Dario Rethage, Jordi Pons, and Xavier Serra. A wavenet for speech denoising. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[25] Sam T. Roweis. One microphone source separation. In *Advances in Neural Information Processing Systems*, 2001.

[26] P. Smaragdis, C. Fevotte, G. J. Mysore, N. Mohammadiha, and M. Hoffman. Static and dynamic source separation using nonnegative factorizations: A unified view. *IEEE Signal Processing Magazine*, 31(3), 2014.

[27] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Adversarial semi-supervised audio source separation applied to singing voice extraction. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2391–2395. IEEE, 2018.

[28] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. Technical Report 1806.03185, arXiv, 2018.

[29] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. The 2018 signal separation evaluation campaign. In *14th International Conference on Latent Variable Analysis and Signal Separation*, 2018.

[30] Naoya Takahashi and Yuki Mitsufuji. Multi-scale multi-band densenets for audio source separation. In *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 21–25. IEEE, 2017.

[31] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation. Technical Report 1805.02410, arXiv, 2018.

[32] Stefan Uhlich, Franck Giron, and Yuki Mitsufuji. Deep neural network based instrument extraction from music. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[33] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[34] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. Technical Report 1609.03499, arXiv, 2016.

[35] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech and Language Processing*, 14(4): 1462–1469, 2006. URL `https://hal.inria.fr/inria-00544230`.

[36] DeLiang Wang and Guy J. Brown, editors. *Computational Auditory Scene Analysis*. IEEE Press, Piscataway, NJ, 2006.

Appendix

A Architecture of the silent sources detector

We use as input a scattering transform of order 2, computed using the Kymatio package [2] with $J := 8$ wavelets per octave. Coefficients of order 1 are indexed by one frequency f_1 and of order two by f_1 and f_2 with f_2 the frequency of the second order filter. We organize the coefficient in a tensor of dimension (C, F, T) where T is the number of time windows, F is the number of order 1 frequencies. The first channel is composed of order 1 coefficients, while the next ones contains the order two coefficient ordered by f_2. Thanks to this reorganization, we can now use 2D convolutions over the output of the scattering transform. The model is then composed of

- batch normalization,
- $\mathrm{Relu}(\mathrm{Conv2d}(C_{in} = 7, C_{out} = 128, K = 5, S = 1))$,
- $\mathrm{Relu}(\mathrm{Conv2d}(C_{in} = 128, C_{out} = 128, K = 5, S = 1))$,
- $\mathrm{Relu}(\mathrm{Conv2d}(C_{in} = 128, C_{out} = 256, K = 1, S = 1))$,
- $\mathrm{MaxPool2d}(K = 5, S = 2)$,
- batch normalization,
- $\mathrm{Relu}(\mathrm{Conv2d}(C_{in} = 256, C_{out} = 256, K = 5, S = 1))$,
- $\mathrm{Relu}(\mathrm{Conv2d}(C_{in} = 256, C_{out} = 256, K = 5, S = 1))$,
- $\mathrm{Relu}(\mathrm{Conv2d}(C_{in} = 256, C_{out} = 512, K = 1, S = 1))$,
- $\mathrm{MaxPool2d}(K = 5, S = 2)$,
- batch normalization,
- frequency dimension is eliminated with a final convolution of kernel size 14 in the frequency axis and 1 in the time axis with 512 input channels and 1024 channels as output,
- BiLSTM with hidden size of 1024, 2 layers, dropout at 0.18,
- $\mathrm{Conv1d}(C_{in} = 2048, C_{out} = 1024, K = 1, S = 1)$,batch normalization, then Relu,
- $\mathrm{Conv1d}(1024, 4, K = 1, S = 1)$.

B Results for all metrics with boxplots

Reusing the notations from [35], let us take a source $j \in 1, 2, 3, 4$ and introduce P_{s_j} (resp $P_{\mathbf{s}}$) the orthogonal projection on s_j (resp on $\mathrm{Span}(s_1, \ldots, s_4)$). We then take with $\hat{s}_j$ the estimate of source s_j

$$s_{\mathrm{target}} := P_{s_j}(\hat{s}_j) \qquad e_{\mathrm{interf}} := P_{\mathbf{s}}(\hat{s}_j) - P_{s_j}(\hat{s}_j) \qquad e_{\mathrm{artif}} := \hat{s}_j - P_{\mathbf{s}}(\hat{s}_j)$$

We can now define various signal to noise ratio, expressed in decibels (dB): the source to distortion ratio

$$\mathrm{SDR} := 10 \log_{10} \frac{\|s_{\mathrm{target}}\|^2}{\|e_{\mathrm{interf}} + e_{\mathrm{artif}}\|^2},$$

the source to interference ratio

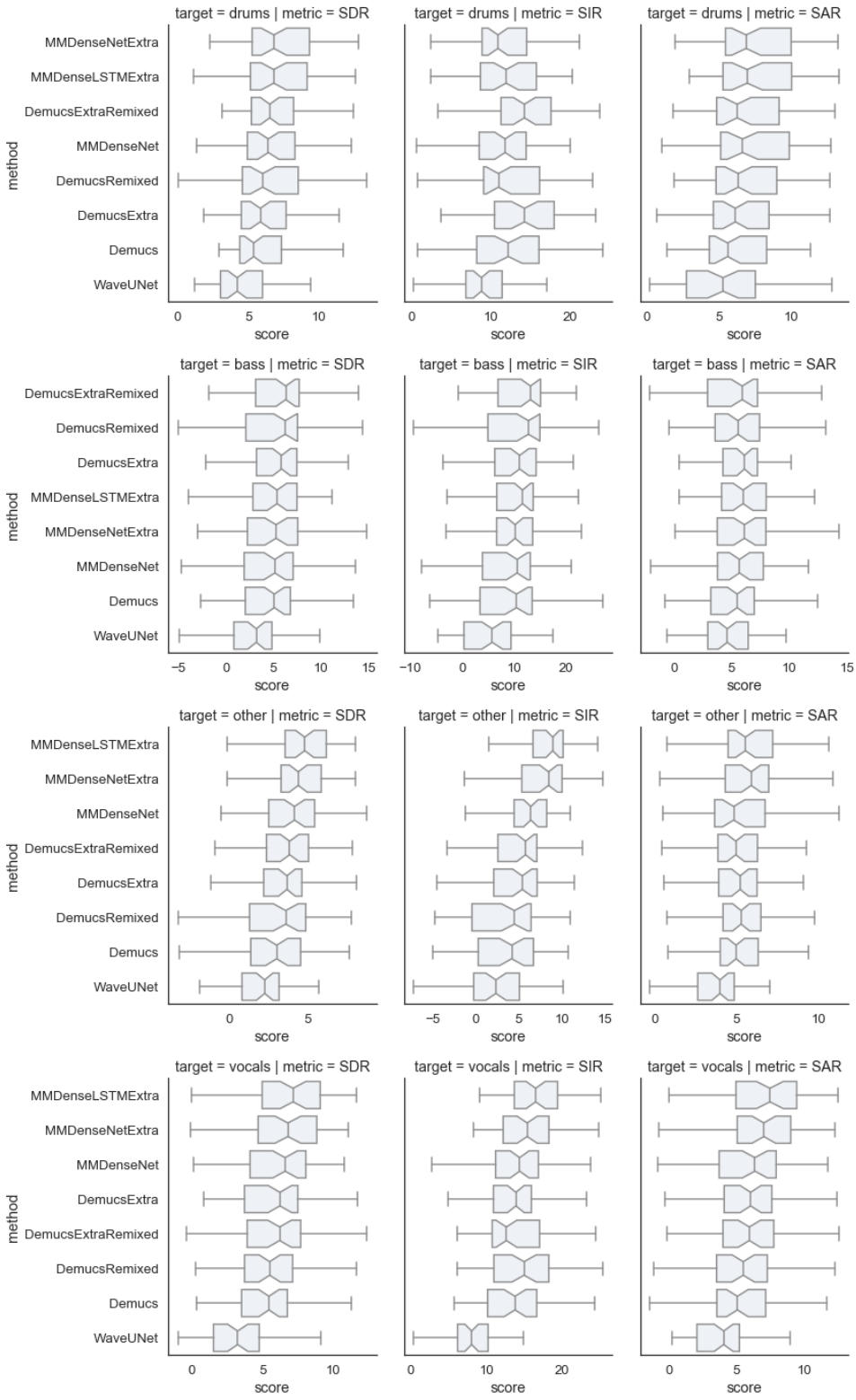
$$\mathrm{SIR} := 10 \log_{10} \frac{\|s_{\mathrm{target}}\|^2}{\|e_{\mathrm{interf}}\|^2}$$

and the sources to artifacts ratio

$$\mathrm{SAR} := 10 \log_{10} \frac{\|s_{\mathrm{target}} + e_{\mathrm{interf}}\|^2}{\|e_{\mathrm{artif}}\|^2}.$$

As explained in the main paper, extra invariants are added when using the `museval` package. We refer the reader to [35] for more details. We provide hereafter box plots for each metric and each target, generated using the notebook provided specifically by the organizers of the SiSec Mus evaluation[5]. An "Extra" suffix means that extra training data has been used and the "Remixed" suffix means that our unlabeled data remixing scheme has been used.

[5] `https://github.com/sigsep/sigsep-mus-2018-analysis`


target = drums | metric = SDR
target = drums | metric = SIR
target = drums | metric = SAR
target = bass | metric = SDR
target = bass | metric = SIR
target = bass | metric = SAR
target = other | metric = SDR
target = other | metric = SIR
target = other | metric = SAR
target = vocals | metric = SDR
target = vocals | metric = SIR
target = vocals | metric = SAR
method
score
MMDenseNetExtra
MMDenseLSTMExtra
DemucsExtraRemixed
MMDenseNet
DemucsRemixed
DemucsExtra
Demucs
WaveUNet