MUSIC SOURCE SEPARATION IN THE WAVEFORM DOMAIN

Anonymous authors
Paper under double-blind review

ABSTRACT

Source separation for music is the task of isolating contributions, or *stems*, from different instruments recorded individually and arranged together to form a song. Such components include voice, bass, drums and any other accompaniments. While end-to-end models that directly generate the waveform are state-of-the-art in many audio synthesis problems, the best multi-instrument source separation models generate masks on the magnitude spectrum and achieve performances far above current end-to-end, waveform-to-waveform models. We present an in-depth analysis of a new architecture, which we will refer to as Demucs, based on a (transposed) convolutional autoencoder, with a bidirectional LSTM at the bottleneck layer and skip-connections as in U-Networks (Ronneberger et al., 2015). Compared to the state-of-the-art waveform-to-waveform model, Wave-U-Net (Stoller et al., 2018), the main features of our approach in addition of the bi-LSTM are the use of transposed convolution layers instead of upsampling-convolution blocks, the use of gated linear units, exponentially growing the number of channels with depth and a new careful initialization of the weights. Results on the MusDB dataset show that our architecture achieves a signal-to-distortion ratio (SDR) nearly 2.2 points higher than the best waveform-to-waveform competitor (from 3.2 to 5.4 SDR). This makes our model match the state-of-the-art performances on this dataset, bridging the performance gap between models that operate on the spectrogram and end-to-end approaches.

1 INTRODUCTION

Cherry first noticed the "cocktail party effect" (Cherry, 1953): 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 (Bregman, 1990) spawned its computational counterpart, trying to reproduce or model accomplishments of the brains with algorithmic means (Wang & Brown, 2006), in particular regarding source separation capabilities.

When producing music, recordings of individual instruments called *stems* are arranged together and mastered into the final song. The goal of source separation is to recover those individual stems from the mixed signal. Unlike the cocktail party 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. In the SiSec Mus evaluation campaign for music separation (Stöter et al., 2018), those individual stems were grouped into 4 broad categories: (1) `drums`, (2) `bass`, (3) `other`, (4) `vocals`. Given a music track which is a mixture of these four sources, also called the mix, the goal is to generate four waveforms that correspond to each of the original sources. We consider here the case of supervised source separation, where the training data contain music tracks (i.e., mixtures), together with the ground truth waveform for each of the sources.

In the fields of speech or music generation, models trained end-to-end to directly synthesize waveforms have outperformed methods that generate spectrograms (Van Den Oord et al., 2016; Mehri et al., 2016; Défossez et al., 2018). However, state-of-the-art approaches in music source separation still operate on the spectrograms generated by the short-time Fourier transform (STFT). They produce a mask on the magnitude spectrums for each frame and each source, and the output audio is generated by running an inverse STFT on the masked spectrograms reusing the input mixture phase

(Takahashi & Mitsufuji, 2017; Takahashi et al., 2018). Several architectures trained end-to-end to directly synthesize the waveforms have been proposed, based on WaveNet-inspired decoders (Lluís et al., 2018), or U-networks (Jansson et al., 2017). They however achieve performances that are far below those of the best approaches that operate on spectrograms: In the last SiSec Mus evaluation campaign (Stöter et al., 2018), the best model that directly predicts waveforms achieves an average signal-to-noise ratio (SDR) over all four sources of 3.2, against 5.3 for the best approach that predicts spectrograms masks (also see Table 1 in Section 5).

In this paper, we propose a new architecture for end-to-end, waveform-to-waveform source separation. The model builds borrows the skip-connections and the encoder/decoder architecture of Wave-U-Net (Jansson et al., 2017), but uses a decoder based on wide transposed convolutions with large strides inspired by recent work on music synthesis (Défossez et al., 2018), rather than the blocks composed of linear interpolation upsampling and convolution layers of Wave-U-Net. The other critical features of the approach are a bidirectional LSTM between the encoder and the decoder, increasing the number of channels exponentially with depth, gated linear units as activation function (Dauphin et al., 2017), and a new initialization scheme. On the MusDB dataset of the last SiSec competition, our approach achieves a state-of-the-art SDR of 5.35 ± 0.03, while being computationally more efficient than the competing methods. In pure terms of performances, it represents an absolute gain of 2.1 SDR compared to the closest end-to-end competitor.

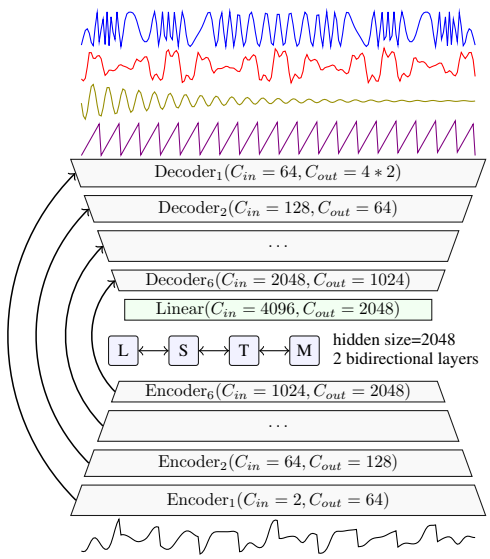
We discuss in more detail the related work in the next Section. We then describe the model and motivate the key components in Section 3. We present the experimental protocol in Section 4, and the experimental results compared to the state-of-the-art in Section 5. We finally report the results of an in-depth ablation study that analyzes the importance of our main design choices.

2 Related Work

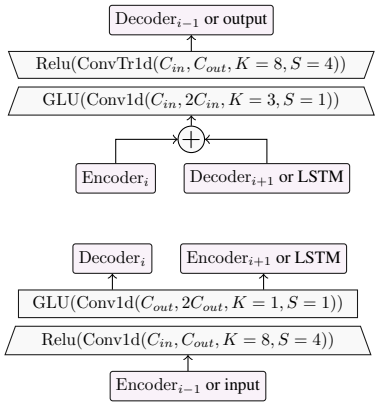
A first category of methods for supervised music source separation work on power spectrograms. They predict a power spectrogram for each source and reuse the phase from the input mixture to synthesise individual waveforms. Traditional methods have mostly focused on blind (unsupervised) source separation. Non-negative matrix factorization techniques (Smaragdis et al., 2014) 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 (Hyvärinen et al., 2004) 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 (Roweis, 2001) or segmentation techniques (Bach & Jordan, 2005).

With the development of deep learning, fully supervised methods have gained momentum. Initial work was performed on speech source separation (Grais et al., 2014), followed by works on music using simple fully connected networks over few spectrogram frames (Uhlich et al., 2015), LSTMs (Uhlich et al., 2017), or multi scale convolutional /recurrent networks (Liu & Yang, 2018; Takahashi & Mitsufuji, 2017). Nugraha et al. (2016) showed that Wiener filtering is an efficient post-processing step for spectrogram-based models and it is now used by all top performing models in this category. Those methods have performed the best during the last SiSec 2018 evaluation (Stöter et al., 2018) for source separation on the MusDB (Rafii et al., 2017) dataset. After the evaluation, a reproducible baseline called Open Unmix has been released by Stöter et al. (2019) and matches the top submissions trained only on MusDB. Our model reaches the same performance as Open Unmix when trained on MusDB. MMDenseLSTM, a model proposed by Takahashi et al. (2018) and trained on 807 unreleased songs currently holds the absolute record of SDR in the SiSec campaign. We nearly attain the same SDR when trained with only 150 extra tracks.

More recently, models operating in the waveform domain have been developed, so far with worse performance than those operating in the spectrogram domain. A convolutional network with a U-Net structure called Wave-U-Net was used first on spectrograms (Jansson et al., 2017) and then adapted to the waveform domain (Stoller et al., 2018). Wave-U-Net was submitted to the SiSec 2018 evaluation campaign with a performance inferior to that of most spectrogram domain models by a large margin. A Wavenet-inspired, although using a regression loss and not auto-regressive, was first used for speech denoising (Rethage et al., 2018) and then adapted to source separation (Lluís et al., 2018). Our model significantly outperforms Wave-U-Net.Given that the Wavenet inspired model performed worse than Wave-U-Net, we did not consider it for comparison.



(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, extra convolutions with kernel 1 (resp 3) in the encoder (resp decoder), the bidirectional LSTM in-between and doubling the channels between layers, allowed by the stride of 4 in all convolutions.

The worse performance of waveform based models might come as a surprise, given the success of architecture like WaveNet (Van Den Oord et al., 2016) or SampleRNN (Mehri et al., 2016). For speech synthesis, good performance was initially obtained with in the spectral domain with Tacotron (Wang et al., 2017) but the synthesis part was later replaced by WaveNet for Tacotron 2 (Shen et al., 2018) with a noticeable reduction of artifacts. Similarly, for music note generation, the work of Engel et al. (2017) on the NSynth dataset showed that end-to-end waveform generation far outperformed spectrogram based methods. Other applications such as domain translation between any instruments was also made possible (Mor et al., 2018). Then how to explain the lagging of end-to-end waveform training for source separation? Unlike for speech or music generation, spectrogram methods in source separation have access to the input signal phase and use it as an approximation of the source phase instead of recovering it with an algorithm like Griffin-Lim (Griffin & Lim, 1984). Besides, most research on the MusDB dataset use a sampling rate of 44 kHz, which makes application of WaveNet computationally prohibitive. In contrast, the NSynth music note dataset is sampled at 16 kHz and WaveNet already requires 32 GPUs for 10 days to be trained on it(Engel et al., 2017). Due to its auto-regressive nature, evaluation is even slower, unless one trains a distilled Parallel WaveNet model (Oord et al., 2017), further lengthening training. An alternative was proposed in the work of Défossez et al. (2018) on the NSynth dataset and achieved higher perceptual quality for a fraction of the computational cost of a WaveNet. Our architecture is inspired by their SING model, reusing simple transposed convolutional and recurrent layers with a regression loss.

3 Model Architecture

Each source s is represented by a waveform $x_s \in \mathbb{R}^{C,T}$ where C is the number of channels (1 for mono, 2 for stereo) and T the number of samples of the waveform. The mixture (i.e., music track) is the sum of all sources $x := \sum_{s=1}^{S} x_s$. We aim at training a model g parameterized by θ, such that $g(x) = (g_s(x;\theta))_{s=1}^{S}$, where $g_s(x;\theta)$ is the predicted waveform for source s given x, that minimizes

$$\min_{\theta} \sum_{\mathbf{s} \in \mathcal{D}} \sum_{s=1}^{S} \ell(g_s(x;\theta), x_s) \tag{1}$$

for some dataset $\mathcal{D}$ and reconstruction error ℓ. The architecture we propose is described in the next few subsections, and the reconstruction loss is discussed in Section 3.3.

The model g we present in this work, which we will refer to as Demucs, takes a stereo mixture as input and outputs a stereo estimate for each source ($C = 2$). It is an encoder/decoder architecture composed of a convolutional encoder, a bidirectional LSTM, and a convolutional decoder, with the encoder and decoder linked with skip U-Net connections. Similarly to other work in generation in both image (Karras et al., 2018; 2017) and sound (Défossez et al., 2018), we do not use batch normalization (Ioffe & Szegedy, 2015) as our early experiments showed that it was detrimental to the model performance. The overall architecture is depicted in Figure 1a.

3.1 Convolutional encoder

The encoder is composed of $L := 6$ stacked convolutional blocks numbered from 1 to L. Each block i is composed of a convolution with kernel width 8, stride 4, C_{i-1} input channels, C_i output channels and ReLU activation, followed by a convolution with kernel size 1, $2C_i$ output channels and gated linear units (GLU) as activation function (Dauphin et al., 2017). Since GLUs halve the number of channels, the final output of block i has C_i output channels. A block is described in Figure 1b.

Convolutions with kernel width 1 increase the depth and expressivity of the model at low computational cost. As we show in our ablation study 5.2, the usage of GLU activations after these convolutions significantly boost performance.

The number of channels in the input mixture is $C_0 = C = 2$, while we use $C_1 := 64$ as the number of output channels for the first encoder block. The number of channels is then doubled at each subsequent block, i.e., $C_i := 2C_{i-1}$ for $i = 2..L$, so the final number of channels is $C_L = 2048$. 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 linear layer to take that number down to C_L.

3.2 Convolutional decoder

The decoder is almost the symmetric of the encoder. It is composed of L blocks numbered in reverse order from L to 1. The i-th blocks starts with a convolution with stride 1 and kernel width 3 to provide context about adjacent time steps, input/output channels C_i and a ReLU activation. Finally, we use a transposed convolution with kernel width 8 and stride 4, C_{i-1} outputs and ReLU activation.

The S sources are synthesized at the final layer only, after all decoder blocks. The final layer is linear with $S.C_0$ output channels, one for each source (4 stereo channels in our case), without any additional activation function. Each of these channels directly generate the corresponding waveform.

U-network structure Similarly to Wave-U-Net (Jansson et al., 2017), the symmetry of the encoder/decoder blocks there are skip connections between the encoder and decoder blocks with the same index, as originally proposed in U-networks (Ronneberger et al., 2015). While the main motivation comes from empirical performances, an immediate advantage of the skip connections is to give a direct access to the original signal, and in particular allows to directly transfers the phase of the input signal to the output, as discussed in Section 3.3.

Motivation: synthesis vs filtering The approach we follow uses the U-Network architecture (Ronneberger et al., 2015), and builds on transposed convolutions with large number of channels and large strides (4) inspired by the approach to the synthesis of music notes of Défossez et al. (2018). The U-net approach was previously used in the context of music source separation in Wave-U-Net (Stoller et al., 2018; Jansson et al., 2017). The fundamental difference of the decoding step between Wave-U-Net and our approach is that Wave-U-Net uses blocks of linear upsampling layers (that double the sampling rate of the signal), followed by convolutions of stride 1. Thus, Wave-U-Net follows a *filtering* approach: it generates its output by iteratively upsampling, adding back the high frequency part of the signal using skip-connections, and filtering with the convolution. We rather follow a direct *synthesis* approach with transposed convolutions that can directly generate a signal at the desired frequency. For the same increase in the number of time steps, our method requires 4 times less operations. Furthermore, Wave-U-Net requires using stride of 1 in the encoder convolutions, as

their output will be used for the skip connections. This further slow down training and also increases memory usage. Those points are verified experimentally in Section 5.2.

3.3 LOSS FUNCTION

For the reconstruction loss $L(g_s(x;\theta), x_s)$ in equation 1, we either use the average mean square error or average absolute error between waveforms: for a waveform x_s containing T samples and corresponding to source s, a predicted waveform $\hat{x}_s$ and denoting with a subscript t the t-th sample of a waveform, we use one of L_1 or L_2:

$$L_1(\hat{x}_s, x_s) = \frac{1}{T}\sum_{t=1}^{T} |\hat{x}_{s,t} - x_{s,t}| \qquad L_2(\hat{x}_s, x_s) = \frac{1}{T}\sum_{t=1}^{T} (\hat{x}_{s,t} - x_{s,t})^2 \,. \tag{2}$$

In generative models for audio, direct reconstruction losses on waveforms can pose difficulties because they are sensitive to the initial phases of the signals: two signals whose only difference is a shift in the initial phase are perceptually the same, but can have arbitrarily high L_1 or L_2 losses. It can be a problem in pure generation tasks because the initial phase of the signal is unknown, and losses on power/magnitude spectrograms are alternative that do not suffer from this lack of specification of the output. Approaches that follow this line either generate spectrograms (e.g., Wang et al., 2017), or use a loss that compares power spectrograms of target/generated waveforms (Défossez et al., 2018).

The problem of invariance to a shift of phase is not as severe in source separation as it is in unconditional generation, because the model has access to the original phase of the signal. This can easily be recovered from the skip connections in U-net-style architectures for separation, and is directly used as input of the inverse STFT for methods that generate masks on power spectrograms. As such, losses such as $L1/L2$ are totally valid for source separation. Early experiments with an additional term including the loss of Défossez et al. (2018) did not suggest that it boosts performance, so we did not pursue this direction any further. Most our experiments use $L1$ loss, and the ablation study presented in Section 5.2 suggests that there is no significant difference between $L1$ and $L2$.

3.4 WEIGHT RESCALING AT INITIALIZATION

The initialization of deep neural networks is known to have a critical impact on the overall performances (Glorot & Bengio, 2010; He et al., 2015), up to the point that Zhang et al. (2019) showed that with a different initialization called fixup, very deep residual networks and transformers can be trained without batch normalization. While Fixup is not designed for U-Net-style skip connections, we observed that the following different initialisation scheme had great positive impact on performances compared to the standard initialization of He et al. (2015) used in U-Networks.

Considering the so-called Kaiming initialization (He et al., 2015) as a baseline, let us look at a single convolution layer for which we denote w the weights after the first initialization. We take $\alpha := \mathrm{std}(w)/a$, where a is a reference scale, and replace w by $w' = w/\sqrt{\alpha}$. Since the original weights have element-wise order of magnitude $(KC_{\mathrm{in}})^{-1/2}$ where K is the kernel width and C_{in} the number of output channels, it means that our initialization scheme produces weights of order of magnitude $(KC_{\mathrm{in}})^{-1/4}$, together with a non-trivial scale. Based on preliminary experiments, we set $a = 0.1$ for all the regular and transposed convolutions.

4 EXPERIMENTAL SETUP

4.1 EVALUATION FRAMEWORK

MusDB and unsupervised datasets We use the MusDB dataset (Rafii et al., 2017) , 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, the mixture, is the sum of those four parts. The first 84 songs form the *train set*, the next 16 songs form the *valid set*[1] while the remaining 50 are kept for the *test set*.

[1]We use the official MusDB python package `https://github.com/sigsep/sigsep-mus-db`.

We collected raw stems for 150 tracks, i.e., individual instrument recordings used in music production software to make a song. We manually assigned each instrument to one of the sources using simple rules on the filenames or listening to the stems in case of ambiguity. We call this extra supervised data the *stem set*. As some of the baselines used additional labeled data (807 songs), we also provide metrics for our own architecture trained using this extra stem set.

Source separation metrics Measurements of the performance of source separation models was developed by Vincent et al. for blind source separation (Vincent et al., 2006) and reused for supervised source separation in the SiSec Mus evaluation campaign (Stöter et al., 2018). Reusing the notations from (Vincent et al., 2006), 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

$$\mathrm{SDR} := 10 \log_{10} \frac{\|s_{\text{target}}\|^2}{\|e_{\text{interf}} + e_{\text{artif}}\|^2}. \tag{3}$$

Note that this definition is invariant to the scaling of $\hat{s}_j$. We used the python package `museval`[2] 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 (Vincent et al., 2006). 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 (Stoller et al., 2018; Takahashi & Mitsufuji, 2017; Takahashi et al., 2018), we focus in this section on the SDR, as it summarizes the best the overall performance of a model but other metrics can be defined (SIR an SAR) and we present them in the supplementary material.

4.2 Baselines

As baselines, we selected Open Unmix (Stöter et al., 2019)[3], a 3-layer BiLSTM model with encoding and decoding fully connected layers on spectrogram frames. It was release by the organizers of the SiSec 2018 to act as a strong reproducible baseline and matches the performances of the best candidates trained only on MusDB. We also selected MMDenseLSTM (Takahashi et al., 2018), a multi-band dense net with LSTMs at different scales of the encoder and decoder. This model was submitted as TAK2 and trained with 804 extra labeled songs[4]. Both MMDenseLSTM and Open Unmix use Wiener filtering (Nugraha et al., 2016) as a last post processing step. The only waveform based method submitted to the evaluation campaign is Wave-U-Net (Stoller et al., 2018) with the identifier STL2. Metrics were downloaded from the SiSec submission repository[5] for Wave-U-Net and MMDenseLSTM. For Open Unmix they were provided by their authors[6].

4.3 Training procedure

We define one epoch over the dataset as a pass over all 11-second extracts with a stride of 1 seconds. We use a random audio shift between 0 and 1 second and keep 10 seconds of audio from there as a training example. We perform the following data augmentation (Uhlich et al., 2017), also used by Open Unmix and MMDenseLSTM: shuffling sources within one batch to generate one new mix, randomly swapping channels. We additionnaly multiply each source by ± 1.

The Demucs separation model described in Section 3 is trained for 240 epochs on 16 Volta GPUs with 32GB or RAM, with a batch size of 128. The learning rate was chosen among [1e-4, 5e-4, 1e-3] and the initial number of channels was chosen in [32, 48, 64] based on the L1 loss on the validation set. Given the cost of training those models, we computed confidence intervals using 5 random seeds in Table 1. For the ablation study on Table 2, we provide metrics for a single run.

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

[3] Reference implementation available at `https://github.com/sigsep/open-unmix-pytorch`.

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

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

[6] `https://zenodo.org/record/3370486`

Table 1: Comparison of our architecture Demucs with the state-of-the-art in the waveform domain (Wave-U-Net) and in the spectrogram domain (Open-Unmix without extra data, MMDenseLSTM with extra data) on the MusDB test set. The *Extra?* indicates the number of extra training songs used. We report the median over all tracks of the median SDR over each track, as done in the SiSec Mus evaluation campaign (Stöter et al., 2018). The `All` column reports the average over all sources. Demucs metrics are averaged over 5 runs, the confidence interval is the standard deviation over $\sqrt{5}$. In bold are the values that are statistically state-of-the-art either with or without extra training data.

			Test SDR in dB				
Architecture	**Wav?**	**Extra?**	`All`	`Drums`	`Bass`	`Other`	`Vocals`
Open-Unmix	✗	✗	**5.33**	5.73	5.23	**4.02**	**6.32**
Wave-U-Net	✓	✗	3.23	4.22	3.21	2.25	3.25
Demucs	✓	✗	**5.35** $\pm.03$	**5.87** $\pm.04$	**5.76** $\pm.04$	3.76 $\pm.05$	6.02 $\pm.05$
Demucs	✓	150	5.97 $\pm.05$	**6.93** $\pm.11$	**6.34** $\pm.07$	3.97$\pm.03$	6.65 $\pm.02$
MMDenseLSTM	✗	804	**6.04**	6.81	5.40	**4.80**	**7.16**

5 Experimental results

In this section we provide here experimental results on the MusDB dataset for our architecture Demucs compared with state-of-the-art baselines and then dive into individual contributions from our architecture specificities detailed in Section 3.

5.1 Comparison with baselines

We provide a comparison the state-of-the-art baselines on Table 1. The models on the top half were trained without any extra data while the lower half used unreleased training songs.We did observe a high variance across runs, sadly no previous work included confidence intervals. As a result, we considered the single metric provided by previous work the exact estimate of their mean performance. We would encourage all future work to include such intervals for easier comparison between models.

Quality of the separation We significantly improved the metrics compared to Wave-U-Net for all sources and matched the overall performance of Open-Unmix when trained with no extra data. Similarly, when trained with only 150 extra songs against 804 for MMDenseLSTM, we achieved similar overall performance than the latter. Our model performs significantly better on the `drums` and `bass` sources while still behind for the `other` and `vocals`. This is the first case of end-to-end waveform models to beat spectrogram domain methods for source separation in music, albeit only for some sources. We provide results for the other metrics (SIR and SAR) as well as box plots with quantiles over the test set tracks in the Appendix, Section B. Audio samples for Demucs and all baselines are provided in the ICLR link code, with more details given in the Appendix, Section A.

Training speed We measured the time taken to process a single batch of size 16 with 10 seconds of audio at 44kHz (the original Wave-U-Net being only trained on 22 kHz audio, we double the time for fairness), ignoring data loading and using `torch.cuda.synchronize` to wait on all kernels to be completed. MMDenseLSTM does not provide a reference implementation. Wave-U-Net takes 1.2 seconds per batch, Open Unmix 0.2 seconds per batch and Demucs 0.9 seconds per batch.

5.2 Ablation study

We provide an ablation study of the main novelties of this paper on Table 2. Given the cost of training a single model, we did not compute confidence intervals for each variation. Yet, any difference inferior to .06, which is the standard deviation observed over 5 repetitions of the Reference model, could be attributed to noise. As mentioned in Section 3.3, we observe a small but non significant improvement when using an L1 loss instead of the MSE loss. Adding a BiLSTM and using the initial weight rescaling described in Section 3.4 provides significant gain, each of around 0.4 points of SDR.

Table 2: Ablation study for the novel elements in our architecture described in Section 3. We use only the train set from MusDB and report best L1 loss over the valid set throughout training as well the SDR on the test set for the epoch that achieved this loss. For the 12 layers model, we increase the number of channels by a factor of $\sqrt{2}$ (rounded to the nearest integer) to match the number of channels of the 6 layers model. Experiments are ordered by increasing Test SDR.

Difference	**Valid set** L1 loss	**Test set** SDR
upsample-convolution instead of transposed	0.166	out of memory
ReLU instead of GLU	0.169	4.62
depth 12, kernel=4, stride=2, $C_{i+1} := \lfloor \sqrt{2} C_i \rfloor$	0.163	4.85
no BiLSTM	0.169	4.86
no initial weight rescaling	0.162	4.88
no convolution with kernel 1 in encoder	0.162	5.04
kernel size of 1 in decoder convolution	0.162	5.16
MSE loss	N/A	5.22
Reference	0.160	5.27

GLUs, extra convolutions in encoder/decoder We introduced extra convolutions in the encoder and decoder, as described in Sections 3.1 and 3.2. The two proved useful, improving the expressivity of the model, especially when combined with GLU activation (Dauphin et al., 2017). Using a kernel size of 3 instead of 1 in the decoder further improves performance. We conjecture that the context from adjacent time steps helps the output of the transposed convolutions to be consistent through time and reduces potential artifacts arising from using a stride of 4.

Upsampling-convolution Those artifacts caused Stoller et al. (2018) to replace transposed convolutions with upsampling followed by convolutions with a stride of 1. We again compared the two approaches. When using upsampling, the convolution in each encoder layer must be performed with a stride of 1 which is merged with a skip connection to the upsampled output of each decoder layer. It allows to recover high frequencies, but the downside is an increased memory usage. The overall number of examples processed per seconds in training is halved compared to our approach using transposed convolutions. We obtain a worse validation loss than our Reference model, and we failed to evaluate the model on the test set as it ran out of memory for one of the track, despite using 32GB GPUs. Splitting the input would be a possibility, but could introduce artifacts at the boundaries (Stoller et al., 2018), especially with the addition of the BiLSTM. Taking $C_1 = 32$ instead of 64 allows to complete the evaluation but with sub-optimal performance (valid loss is 0.171, SDR 4.24). Thus, transposed convolutions achieve higher quality, run faster and have a smaller memory footprint.

Model depth We also tried training a model with a depth of 12, a kernel size of 4 and stride of 2, increasing the channels by $\sqrt{2}$ for fairness with the Reference model. Training diverged with a learning rate of 5e-4, so we used 1e-4 and doubled the number of epochs. Training speed is largely impaired (92 examples/sec on 16 GPUs against 211) as well as the final training loss. We conclude that our approach works best with a "stocky and short" rather than a "deep and thin" architecture.

CONCLUSION

We presented Demucs, a simple architecture inspired by previous work in source separation from the waveform and audio synthesis that bridges the gap between spectrogram and waveform based methods, even significantly improving on them for the `drums` and `bass` sources, while increasing the SDR compared to best performing waveform domain Wave-U-Net (Stoller et al., 2018) by 2.2 points. We provided a comprehensive analysis of the differences between our architecture and Wave-U-Net, showing how to combine simple convolutions, transposed convolutions, BiLSTM and GLU activations to achieve state-of-the-art quality. We hope our architecture can provide a useful foundation for solving other audio tasks requiring analysis and synthesis, especially when using more computationally intensive alternatives like WaveNet (Van Den Oord et al., 2016) or SampleRNN (Mehri et al., 2016) is not feasible.

REFERENCES

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

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

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.

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.

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.

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.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

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.

Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1984.

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.

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

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

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.

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.

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

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.

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.

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.

Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. Technical Report 1805.07848, arXiv, 2018.

Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. Multichannel music separation with deep neural networks. In *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. Technical Report 1711.10433, arXiv, 2017.

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

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.

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.

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

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

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.

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.

F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji. Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software*, 2019.

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.

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)*. IEEE, 2017.

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.

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.

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.

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.

Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech and Language Processing*, 2006.

DeLiang Wang and Guy J. Brown (eds.). *Computational Auditory Scene Analysis*. IEEE Press, Piscataway, NJ, 2006.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. Technical Report 1703.10135, arXiv, 2017.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

APPENDIX

A AUDIO SAMPLES

We provide audio samples taken from the test set of MusDB. They are available through the ICLR code sharing url[7]. To download the archive, click on "Download" > "Direct Download" in the top right of the page. The audio files for the Wave-U-Net and MMDenseLSTM have been obtained from the SiSec Mus 2018 evaluation campaign results website[8]. For Open Unmix, we generated them from the pretrained UMX model using the reference PyTorch implementation[9]. We recommend listening to the audio samples with headphones, while being careful with the volume. An HTML page `index.html` is provided for easier comparison. The following folders are provided:

- Reference: ground truth,
- Open Unmix,
- WaveUNet,
- Demucs: trained only on MusDB,
- DemucsExtra: trained on MusDB and an extra 150 songs.
- MMDenseNetLSTM, trained on MusDB and an extra 804 songs,

B RESULTS FOR ALL METRICS WITH BOX PLOTS

Reusing the notations from Vincent et al. (2006), 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

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

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

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

the source to interference ratio

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

and the sources to artifacts ratio

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

As explained in the main paper, extra invariants are added when using the `museval` package. We refer the reader to Vincent et al. (2006) for more details. We provide box plots for each metric and each target on Figure 2, generated using the notebook provided specifically by the organizers of the SiSec Mus evaluation campaign[10]. Hereafter, we provide the equivalent of Table 1 in the main paper for both SIR and SAR.

[7] `https://www.dropbox.com/sh/o0gps94s120v7l4/AABS5vDfuuRjgY_zDjdSm_Fsa?dl=0`

[8] `https://sisec18.unmix.app`

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

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

			Test SIR in dB				
Architecture	**Wav?**	**Extra?**	All	Drums	Bass	Other	Vocals
Open-Unmix	✗	✗	**10.49**	11.12	**10.93**	**6.59**	13.33
Wave-U-Net	✓	✗	6.26	8.83	5.78	2.37	8.06
Demucs	✓	✗	10.34 ±.08	**11.93** ±.22	9.97 ±.27	5.34 ±.10	**14.13** ±.20
Demucs	✓	150	11.59 ±.08	**13.23** ±.12	**12.59** ±.18	6.10±.16	14.45 ±.16
MMDenseLSTM	✗	804	**12.24**	11.94	11.59	**8.94**	**16.48**

			Test SAR in dB				
Architecture	**Wav?**	**Extra?**	All	Drums	Bass	Other	Vocals
Open-Unmix	✗	✗	**5.90**	**6.02**	**6.34**	4.74	**6.52**
Wave-U-Net	✓	✗	4.49	5.29	4.64	3.99	4.05
Demucs	✓	✗	5.76 ±.04	**5.99** ±.05	6.07 ±.09	**5.02** ±.06	5.96 ±.07
Demucs	✓	150	5.97 ±.05	6.04 ±.04	**6.65** ±.08	4.92±.03	6.44 ±.07
MMDenseLSTM	✗	804	**6.50**	6.96	6.00	**5.55**	**7.48**

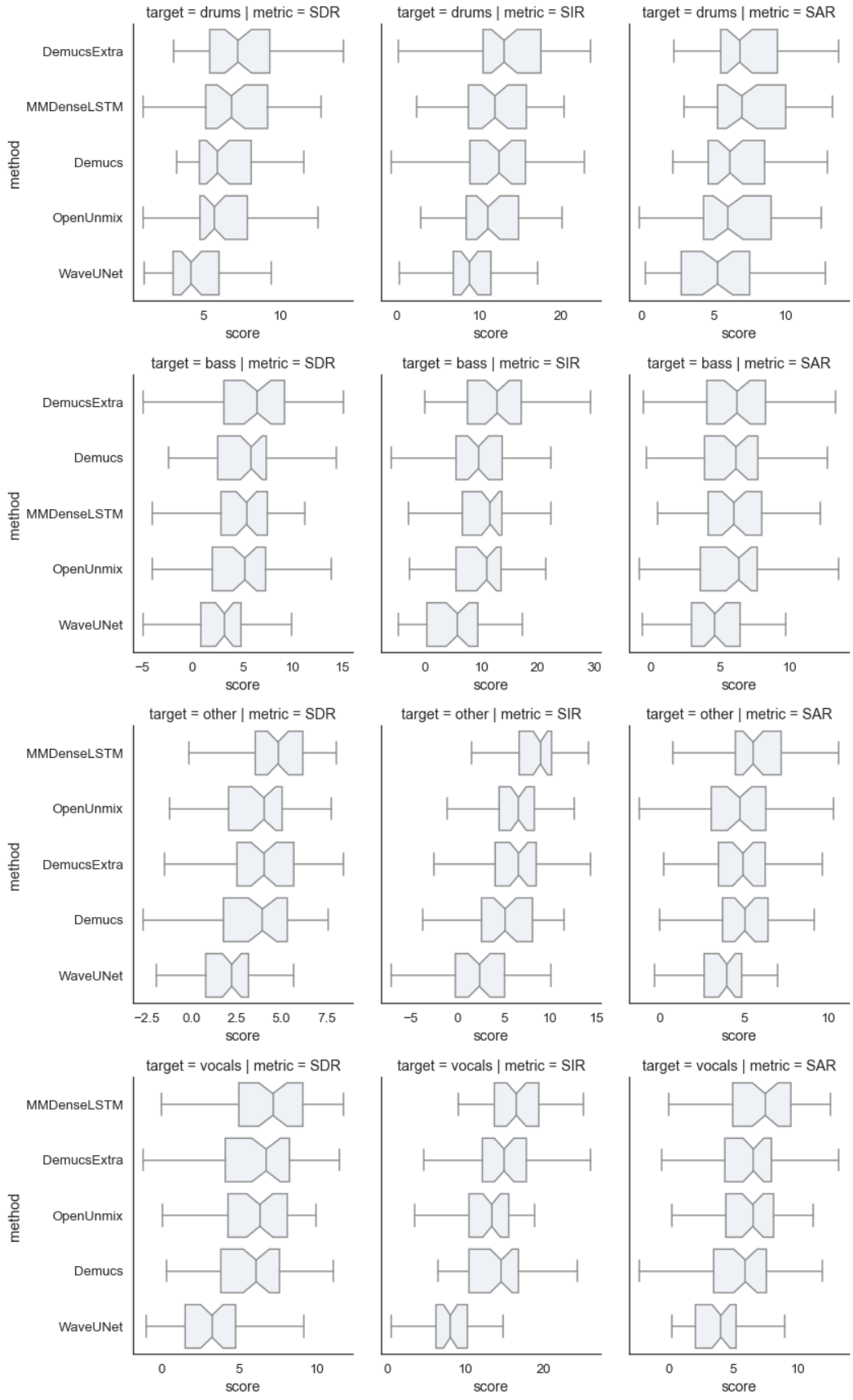


Figure 2: Boxplot showing the distribution of SDR, SIR and SAR over the tracks of the MusDB test.