Monaural Music Source Separation using a ResNet Latent Separator Network

Gino Brunner, Nawel Naas, Sveinn Palsson, Oliver Richter, Roger Wattenhofer[†]
Departement of Electrical Engineering and Information Technology
ETH Zurich
Zurich, Switzerland
{brunnegi,naasn,spalsson,richtero,wattenhofer}@ethz.ch

***Abstract*—In this paper we study the problem of monaural music source separation, where a piece of music is to be separated into its main constituent sources. We propose a simple yet effective deep neural network architecture based on a ResNet autoencoder. We investigate several data augmentation and post-processing methods to improve the separation results and outperform various state of the art monaural source separation methods on the DSD100 and MUSDB18 datasets. Our results suggest that in order to further push the state of the art in monaural music source separation we need more data, better data augmentation methods, as well as more effective post-processing methods; and not necessarily ever more complex neural network architectures.**

I. Introduction

Most people are familiar with the so-called cocktail party problem, where a number of people are talking simultaneously, but one only wants to follow a particular conversation. Such a *source separation* is a classic problem for humans as well as in digital signal processing: Given a mixture of signals, the objective is to recover its original components. The human ears record each source in stereo with a short time offset (depending on the angle of the source), such that stereo source separation is trivial for most humans [1].

Source separation becomes more difficult, for humans as well as machines, if we only have a *single* input channel (a monaural or monophonic signal, often just mono), since the valuable spatial information is missing in the input. While nowadays many audio recordings are in stereo, there are still numerous exceptions. For example, most smartphones and many digital cameras only have a mono microphone. These undirected microphones are often of relatively low quality and thus recording fidelity suffers, especially in noisy environments. Monaural audio source separation can help here, in that it separates the signal of interest from the remainder, e.g., voice from noise. We focus on music source separation, an important problem on its own with many applications such as automatic music transcription, lyrics recognition and pitch estimation.

Recently, data-driven methods (in particular deep neural networks) have outperformed classical methods in many domains, including music source separation. One of the advantages of deep learning is that salient features are automatically learned, alleviating the need for feature-engineering. In this paper we exploit this end-to-end learning capability for monaural music source separation. In particular, we aim to separate monaural music into four sources: singing voice, bass, drums, and other instruments. The caveat with deep learning methods is that one is tempted to build complex network architectures, giving rise to vague explanations of why some particular architecture might work. But in essence, deep neural networks are general function approximators, often requiring only little tuning in terms of architecture hyper parameters. Along this line we show that a simple architecture can improve upon the state of the art in monaural music source separation. We further provide insights into data augmentation and post processing methods and introduce a new post processing method based on energy thresholding. In order to facilitate future research and promote reproducibility, we open source our code.[1]

II. Related work

Audio source separation is a fundamental problem in signal processing and has been studied for well over 50 years. Before the advent of deep learning methods some of the most widely used techniques included non-negative matrix factorization (NMF), independent component analysis (ICA) or probabilistic latent component analysis (PLCA). Recently, data-driven methods, in particular methods based on deep learning, mostly outperformed non data-driven approaches. This trend is exemplified by the latest Signal Separation Evaluation Campaigns (SiSEC) [2], [3] where deep learning based methods achieved significantly higher scores. The body of existing work is extensive, and hence we focus only on recent deep learning approaches. For a more comprehensive overview we refer the interested reader to [4], [5].

Most deep learning approaches are based on supervised learning, that is, they make use of labeled data. Supervised methods can further be split into single-channel (monaural) and multi-channel (usually stereo) methods. In this work we focus on monaural source separation. One of the earliest applications of deep learning to monaural source separation was introduced in [6]. In this work, Huang et al. use feed-forward and recurrent neural networks (RNNs) to output time-frequency masks (TF masks) for each source, which are then

[†] Authors listed in alphabetical order

[1] https://github.com/sveinnpalsson/sourceseparation

applied to the spectrogram of the original mixture to perform source separation. Using a TF mask to constrain the sum of predicted sources to equal the original mixture is a technique commonly used by model and learning based source separation methods and we adapt this idea in some of our experiments. In [7], they further improve their method and apply it to additional tasks such as singing voice separation and speech noise reduction. Miron et al. [8] combine TF masking with convolutional neural networks (CNNs) to separate sources in classical music. Chandna et al. [9] show the advantages of using CNNs instead of multi-layer perceptrons (MLPs) in an autoencoder based architecture, both in terms of resulting signal quality and runtime. Mimilakis et al. [10] introduce an approach based on RNNs that learns the parameters of the TF mask computation, instead of specifying it as a deterministic function, which helps reduce interference between sources. Drossos et al. [11] additionally regularize the RNN to improve the learning of long-term dependencies, and achieve state of the art results in singing voice separation. While we focus on separating not only voices but also the remaining instruments, we nevertheless achieve better results on singing voice separation. The state of the art in monaural music source separation was introduced by Li et al. [12]. They proposed a complex model that first uses an encoder based on CNNs and RNNs to learn latent features, and then an MLP to perform source decoupling in the latent space before using a CNN decoder to reconstruct the separated sources. In contrast to [12], we propose a simpler, ResNet [13] based, autoencoder model that still achieves source separation in the latent space. Our quantitative results are superior to [12]. In contrast to [12], we also provide audio samples to show the limitations of our method.

While we specifically target mono source separation, there is a substantial amount of successful work in the stereo domain. The current state of the art is achieved by a family of models based on densely connected CNNs [14], [15]. Park et al. [16] introduce a method based on stacked hourglass networks and achieve results comparable to state of the art. Stoller et al. [17] introduce an architecture based on U-Net [18] and show that it is possible to directly work on raw audio waves without performing an explicit fourier transform. Note that source separation from a stereo signal is easier due to the spatial information available. We therefore do not directly compare against these methods.

III. Dataset and Performance Metrics

We evaluate our methods on the DSD100 [2] and MUSDB18 [19] datasets. DSD100 contains 100 professionally produced recordings with a pre-defined train/test split of 50/50. MUSDB18 is a superset of DSD100 with 50 additional recordings and a train/test split of 100/50. Each recording consists of 4 stereo source signals: vocals, drums, bass and other. The *other* category contains a mixture of everything that is not vocals, bass or drums. We convert the audio samples to mono by averaging both input channels. To evaluate source separation performance, the current best practice is to report the signal-to-distortion ratio (SDR), signal-to-interference ratio (SIR) and signal-to-artifact ratio (SAR), which can be calculated as follows (see [20] for details):

$$\begin{aligned}\text{SDR} &= 10\log_{10}(||s_{target}||^2/||e_{interf}+e_{noise}+e_{artif}||^2)\\ \text{SIR} &= 10\log_{10}(||s_{target}||^2/||e_{interf}||^2)\\ \text{SAR} &= 10\log_{10}(||s_{target}+e_{interf}+e_{noise}||^2/||e_{artif}||^2)\end{aligned}$$

where $s_{target} = \langle \hat{s}, s\rangle s/||s||^2$. s and $\hat{s}$ are the reference and estimated sources. $e_{interf}, e_{noise}, e_{artif}$ are the distortions corresponding to interference, noise and artifacts, respectively. The most important metric is the SDR, which measures the overall target signal energy actually contained in the source estimate compared to everything else (distortion). It is generally believed that SDR correlates best with human perception of quality. The other metrics, SIR and SAR, measure the presence of other sources and external artifacts in the estimate, respectively. We use the python implementation from [3] to compute the metrics.

Computing these metrics for the whole dataset at once is not computationally feasible. Even for single songs the memory consumption is usually too high. Therefore the test recordings are split into windows of fixed length (e.g., 1s or 30s) and the metrics are then computed for each window. The final result is computed by taking the mean or median over the windows. The median is a more robust measure of performance in this case due to outliers that come from windows that have low signal energy (near silent) where small distortions in the estimate result in high negative SDR values.

IV. Methods

Audio source separation is the process of recovering one or more source signals from an observed mixture signal $x(t) = \sum_{j=1}^{J} s^{(j)}(t)$, where $s^{(j)} \in \mathbb{R}^L$ are the signal sources and L is the number of signal channels. For mono source separation we have $L = 1$.

The most common approach to source separation is to work with signals in the time-frequency domain via the short-time Fourier transform (STFT). We denote the STFT of a signal source as $S^{(j)}(\tau, f) = \text{STFT}(s^{(j)}(t))$, where τ and f are the indices for the time and frequency frames, respectively. Because the STFT is an additive transformation, the mixture signal in the time-frequency domain is

$$X(\tau, f) = \sum_{j=1}^{J} S^{(j)}(\tau, f)$$

A. Architecture

The source separation task can be stated as a supervised learning problem, where we have samples (mixture signals) and targets (separated sources). We train an autoencoder with a ResNet-based [13] latent separation network to estimate the individual sources from the mixture. The architecture of our model is depicted in Figure 1. The idea is to map inputs to a latent space via an encoder and then through a residual

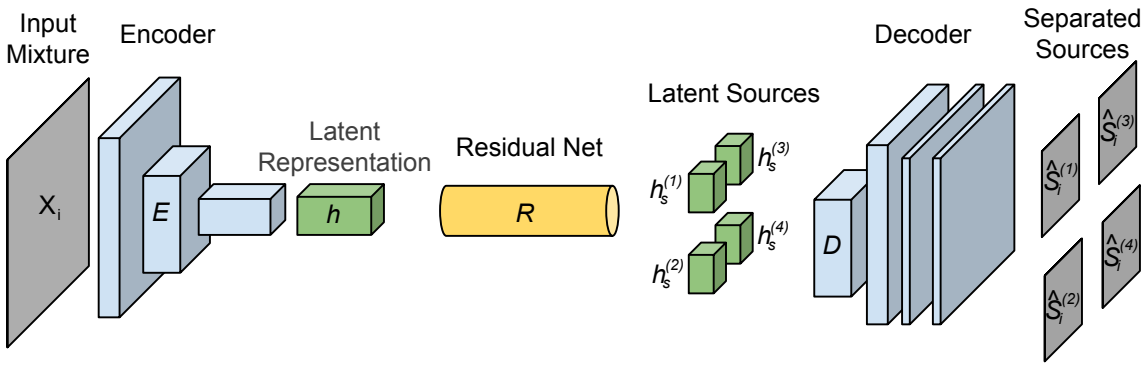


Fig. 1: Architecture of our model. The input is a complex valued spectrogram segment of a mixture signal. The encoder outputs a lower dimensional latent representation. A residual net then processes the latent representation further into 4 separate latent sources, one for each source of the mixture. Each of these separated latent sources is then independently fed to the decoder to produce the estimates of the separated sources.

Part of model	Function	Output dimension	Parameters for $J = 4, K = 256, T = 140, F = 513$
Input Spectogram	X_i	$T \times F \times 2$	
CNN Encoder	$E(X_i) = h$	$\frac{T}{4} \times \frac{F}{4} \times K$	1,486,208
Latent separator (ResNet)	$R(h) = h_s = [h_s^{(0)}, h_s^{(1)}, ..., h_s^{(J)}]$	$J \times \frac{T}{4} \times \frac{F}{4} \times \frac{K}{J}$	7,084,032
CNN Decoder	$\hat{S}_i^{(j)} = D(h_s^{(j)})$	$T \times F \times 2$	605,608

TABLE I: The main parts of our model architecture along with their respective output dimensions. The input to the model is a segment of time-frequency data with T time frames and F frequency bins. The input has 2 channels, corresponding to the real and imaginary parts of the data. We also give an example, in the last column, of the number of parameters for each part of the model, with the configuration we use in our baseline model.

network to J separated latent representations. The resulting separated latent representations are then passed through a shared decoder that reconstructs an estimate of the corresponding source spectograms.

More formally, we segment the input mixture $X(\tau, f)$ along the time axis into segments of length T. Each segment contains F frequency bands, covering the full frequency range. We denote one such segment as X_i, with $i \in \{1, ..., N\}$, where N is the number of segments in the dataset. The mixture signal is complex valued, but can also be thought of as two real valued signals by looking at the real and imaginary part separately. The resulting input dimension of X_i is then $T \times F \times 2$, where the 2 stems from the real and imaginary part of the segment. In this way we also preserve phase information. The input mixture X_i is first passed through the encoder E which creates the first latent representation $h = E(X_i)$. The encoder is a 3-layer CNN with batch normalization and ReLU activations on each layer. The output of the encoder has shape $\frac{T}{4} \times \frac{F}{4} \times K$, where K is a hyper parameter that we discuss later in this section. This first latent representation h is then passed through a residual network R to produce J new latent representations, one for each of the J sources in the mixture, $R(h) = h_s = \left[h_s^{(1)}, ..., h_s^{(J)}\right]$. The dimensions of h_s are the same as h, but along the last dimension we partition the K channels into J segments. Each of the J segments of h_s has shape $\frac{T}{4} \times \frac{F}{4} \times \frac{K}{J}$ and is the latent representation of its corresponding source. The residual network is a ResNet [13] with 6 residual blocks. Now, for each of the $h_s^{(j)}$, the j-th source is estimated by a decoder D as $\hat{S}^{(j)} = D(h_s^{(j)})$. The decoder is a 4-layer CNN with batch normalization and ReLU activations on all layers, except for the output layer. Note that we share the decoder among sources, as we expect the sources to already be separated in the latent space. The decoder's task is simply to map the latent representation into a corresponding spectogram. We also evaluate using four separate specialized decoders instead of one shared decoder (see Section V-E). The experimental results presented in Section V are achieved with a single shared decoder, except for the MUSDB18 experiments, where we used separate decoders.

We train our model to minimize the squared L_2 norm of the difference between the real and estimated sources in the time-frequency domain. Formally the loss function is:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{J} ||S_i^{(j)} - \hat{S}_i{}^{(j)}||^2$$

For the STFT, we compute the 1024-point FFT, using Hann windows of size 1024 that overlap by 75%. Since the input data has a sampling rate of 44100 Hz, the frequency range is 22050 Hz and represented by $F = 513$ frequency bands. We choose the model's time context to be $T = 140$ in our experiments, which corresponds to about 0.8 seconds of

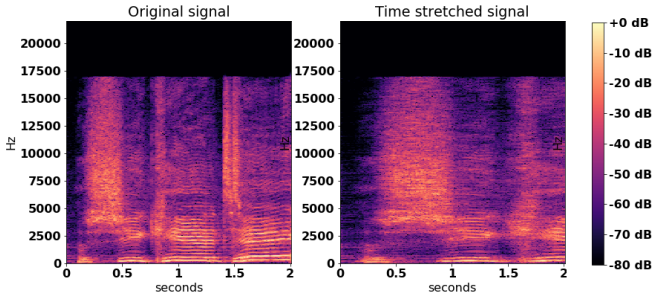


Fig. 2: Original audio signal (left) and the tempo augmented version (right). One can see that the pitch stays unaffected, while the signal is stretched in time.

the original audio waveform for each input sample X_i. For the encoder output depth, we choose $K = 256$. The model is implemented in Tensorflow [21] and the loss function is minimized via the SGD based Adam optimizer [22] with a learning rate of $\alpha = 5 \cdot 10^{-5}$. Table I lists various output dimensions and number of parameters of our architecture. We refer the interested reader to our openly available source code for further implementation details.

B. Time Frequency Masking

Source separation models often contain a post processing step called the time-frequency mask (TF mask). The TF mask ensures that the sum of estimated sources is exactly equal to the original mixture. The TF mask is defined for each source as $M_j = ||\hat{S}^{(j)}||^\alpha / \sum_{j=1}^{J} ||\hat{S}^{(j)}||^\alpha$ and the final estimate of source j is then computed as element-wise multiplication of the input mixture with M_j as $\tilde{S}^{(j)} = X \odot M_j$. In [23], the most appropriate value of α was shown to be around 1, corresponding to amplitude-based masking. Although masking has been found to improve separation quality in some cases, we did not consistently observe positive effects.

C. Data augmentation

In our experiments we initially observed a large generalization error, i.e., our model managed to fit the training data, but then struggled to generalize to unseen samples. This is a typical problem when using high-capacity neural networks, and suggests that more data is required. Since we are restricted to the MUSDB18 dataset and gathering more data is highly non-trivial, we resort to data augmentation to close the generalization gap.

The first augmentation method consists of stretching/compressing the signals in time, i.e., changing the tempo of the music. We provide a visual illustration of this augmentation method in Figure 2. Stretching/compressing could be achieved by changing the sampling rate, but the result would be pitch shifted. A phase vocoder provides a better way of separating temporal and spectral modifications of audio signals. Given an STFT of an audio signal, the time stretched signal can be created by simply applying the inverse FFT with different spacing between FFT windows. This alone would produce phase artifacts, which the phase vocoder resolves by scaling the phase of the reconstructed signal by the same factor as the time expansion (see also [24]). This augmentation method was introduced in [25] in the context of singing voice detection. We sample the stretch factor from the values $[0.5, 0.75, 1, 1.25, 1.5, 1.75, 2]$, whereas 1 (i.e., no stretching) is sampled with probability $\frac{2}{5}$, and the remaining values are sampled uniformly. This choice was made heuristically without any tuning based on the idea that we want to retain more of the unaltered songs relative to the stretched songs.

The second method we employ was proposed in [26] and consists of creating new mixtures by combining sources from different songs. More specifically, we load one batch of original mixture segments and then construct new mixture segments by randomly recombining the sources, where we make sure that the created mixtures contain each source type at most once.

D. Post Processing

Here we describe two post processing methods that can be used to improve the quality of the source separation.

1) Overlapping Input Segments: At test time we apply the model to input audio recordings *sequentially*, and we can thus choose to overlap consecutive segments X_i for more robust estimates. Let γ be the overlap factor, and n the number of time frames in the STFT representation of the input recording. Then the number of segments will be $\frac{1}{(1-\gamma)}\frac{n}{T}$. The overlap can vary from $\gamma = 0$ (no overlap) to $\gamma = \frac{T-1}{T}$ (maximum overlap). The final output spectrogram at a certain time t is computed by averaging all of the overlapping segments at time t. This method was also used in [12] with $\gamma = 0.3$.

2) Signal Energy Thresholding: We note that the output of the model will generally not be exactly zero and always contain some noise or artifacts, even if the ground truth source is actually silent. When listening to the separated sources however, it is often evident when a source should be silent. Hence, it should be easy to improve the sound quality by setting the sources to exactly zero in these cases. We therefore model a source $S_i^{(j)}$ as a continuous valued signal $C_i^{(j)}$ multiplied by a binary signal $P_i^{(j)}$, i.e., $S_i^{(j)} = P_i^{(j)} \cdot C_i^{(j)}$. The binary valued signal $P_i^{(j)}$ is either one, if the source is estimated to be present in the mixture, or zero if not. We can thus use $P_i^{(j)}$ to switch sources on or off. We found that a very simple model of $P_i^{(j)}$ yields better audio quality, both perceptually and in terms of SDR, than applying no thresholding at all. Despite its simplicity, we are unaware of any related work using a similar thresholding post processing step.

Specifically, we propose to estimate the presence of a signal by simply looking at the signal energies, e_j and e_m, corresponding to the energy of the j-th source estimate and the energy of the mixture, respectively:

$$P_i^{(j)} = \begin{cases} 1, & \text{if } e_j \geq t_j \text{ and } e_m \geq t_m \\ 0, & \text{otherwise} \end{cases}$$

Augmentation	Source	SDR	SIR	SAR
None	Vocals	3.98	9.63	6.07
	Drums	3.89	**12.21**	4.96
	Bass	**1.85**	**6.46**	4.78
	Other	0.86	5.31	4.03
Tempo	Vocals	3.68	9.14	6.07
	Drums	3.73	10.63	5.18
	Bass	1.50	5.71	5.04
	Other	0.73	5.43	3.81
Shuffle	Vocals	3.82	8.24	**6.73**
	Drums	3.88	9.66	**5.91**
	Bass	1.65	5.91	**5.19**
	Other	**1.58**	5.92	**4.68**
Shuffle + Tempo	Vocals	**4.57**	**10.04**	6.67
	Drums	**4.28**	11.55	5.59
	Bass	1.84	6.28	5.06
	Other	1.37	**6.25**	4.15

TABLE II: Comparison of augmentation methods on DSD100. Values are in [dB].

γ	SDR	SIR	SAR	sec/song
0	3.24	**9.23**	5.43	**28**
0.95	**3.47**	9.09	**5.86**	300

TABLE III: Effect of using overlapping input segments at test time. SDR, SIR and SAR values are the means over all sources. γ indicates the overlap factor. Values are in [dB].

Method	Source	SDR	SIR	SAR
No thresholding	Vocals	3.98	9.63	**6.07**
	Drums	3.89	12.21	**4.96**
	Bass	1.85	6.46	**4.78**
	Other	0.86	5.31	4.03
With thresholding	Vocals	**4.69**	**11.45**	5.60
	Drums	**4.52**	**13.14**	4.72
	Bass	**2.62**	**6.76**	3.60
	Other	**3.01**	**5.40**	**4.12**

TABLE IV: Effect of signal energy thresholding on DSD100. Values are in [dB].

where t_m and t_j correspond to thresholds that need to be determined. Note that t_m can be chosen individually for each source type. In our experiments, we choose conservative threshold values without any tuning.

Here we want to highlight an innate problem with the commonly used evaluation metrics, especially when calculating the performance metrics with short 1-second windows, as suggested in the SiSec 2018 campaign [3]: When a signal is estimated as silent in a whole 1s window, the result will be $\log(0) = NaN$, and the window will then be excluded from the evaluation. Thus, in order to achieve very high scores one could simply choose high threshold values to exclude all segments with low signal energy, which will also exclude most negative SDR outliers. However, this increase in SDR would likely not reflect audible quality, as the resulting signals would be heavily fragmented, i.e., only a few remaining segments with high signal energy will remain. When using longer evaluation windows of, e.g., 30 seconds, such as in the SiSec 2016 campaign [2], a very high threshold would be needed to exclude the whole window from the evaluation. We therefore report our results for both evaluation methods, to allow a fair comparison to future work. Despite the unintended effect on the metric, we found audible quality to improve as well for our conservative threshold choice. We would like to make an appeal to the research community at this point to not blindly rely on the SDR/SIR/SAR results reported without checking the audible quality.

V. Experiments and Results

We evaluate the performance of our model for different data augmentation methods and post processing steps. We then compare our results directly to current state of the art monaural source separation methods on DSD100 and MUSDB18. Finally, we investigate our architecture choice and hint at a current limitation. All experiments have the following setup unless stated otherwise: We train the models on the training set of the DSD100 or MUSDB18 datasets and report performance on the whole test set. The performance metrics (SDR, SIR and SAR) are computed by matching 30 second long windows, overlapping by 50%, and then computing the median. *NaN* values are excluted from the computation of the median. *NaN* values occur when a reference source is silent in a given window and thus the ratios are undefined. We note that the median is a more robust measure than the mean since the evaluation method is prone to large negative outliers. Such outliers occur when a particular reference signal has very low energy but is estimated with some noise of higher energy, which results in large negative SDR values.

A. Data Augmentation

We train a baseline model with each of the augmentation methods introduced earlier to measure their effects. The results are shown in Table II. Surprisingly, when applying the data augmentation methods individually, only small improvements, or even slight degradation, can be observed. However, when combining both methods, the improvements are quite substantial. Only the bass source does not get a significant boost in SDR. This suggests that data augmentation is important, or correspondingly, larger data sets are needed to train better models.

B. Post Processing Methods

The effect of using overlapping input segments during testing is shown in Table III. Results are shown for evaluating our baseline model with $\gamma = 0$ and $\gamma = 0.95$. Using overlapping windows during testing indeed seems to improve performance, albeit at the cost of increased computation time proportional to the overlap factor.

Table IV shows the results of the energy based thresholding. For all sources we choose

$$t_j = 0.001 \qquad t_m = 0.001$$

except for *other*, where we set

$$t_j = 0.0001 \qquad t_m = 0$$

Method	Source	SDR	SIR	SAR
DRDNN	Vocals	3.15	**9.39**	**7.66**
	Drums	3.23	7.35	**6.59**
	Bass	**3.41**	3.14	**7.17**
	Other	1.87	1.97	**6.81**
RA	Vocals	3.46	8.94	5.89
	Drums	4.63	**12.26**	5.74
	Bass	3.21	**7.60**	5.60
	Other	**2.27**	**4.97**	3.99
RA+th	Vocals	**3.58**	8.87	5.90
	Drums	**4.80**	11.50	5.79
	Bass	3.26	7.29	5.62
	Other	**2.27**	4.97	4.00

TABLE V: Comparison to prior state of the art (DRDNN [12]) on DSD100 with 10-fold cross validation. Values are in [dB].

Method	SDR	SIR
MIM-NINF	3.63	7.06
STO2	3.92	6.75
RA	3.98	9.63
JEO2	4.07	6.09
MIN-RINF	4.2	7.94
MadTwinNet	4.57	8.17
RA+aug	4.57	10.04
RA+aug+mask	4.99	9.37
RA+aug+mask+th	5.66	11.31
RA+aug+mask+th+o	**5.70**	**11.57**

TABLE VI: Singing voice separation of monaural methods on DSD100. Values are in [dB]. The comparison results are taken from [11].

The simple thresholding step substantially improves SDR and SIR, while SAR gets slightly worse for most sources. The additional artifacts likely stem from the sharp transitions in the output spectrogram introduced by the thresholding. Nevertheless, since SDR is generally regarded as the most important metric, this indicates that our thresholding method is effective, which is also reflected in the audible quality of the separated sources.[2]

C. Evaluation on DSD100

For the remainder of the evaluation we show results of multiple versions of our model. The baseline is denoted as *RA* (ResNet Autoencoder). The use of data augmentation (tempo and shuffling) is denoted by *aug*. Time frequency masking (see Section IV-B) is denoted by *mask* and we indicate the use of energy thresholding and overlapping input segments by *th* and *o*, respectively. In this section we show our models' performance along with other recent state-of-the-art methods. The method we compare with for complete source separation is DRDNN [12]. We further compare against multiple methods that are specialized in monaural singing voice separation, including MadTwinNet [11], the current state of the art in this domain.

Li et al. [12] evaluate DRDNN with 10-fold cross validation to obtain more robust performance estimates. That is, they split the training set of DSD100 into 10 folds to train 10 models and then average the results. This approach deviates from the standard evaluation method for DSD100 as stated in [2], where a 50/50 train/test split is recommended. We conducted tailored experiments to match the setup in [12]. However, we did not train our model to its full capacity due to the large computational effort required by this evaluation method. In particular, we trained 10 models for 125,000 steps each, which translates to 720 hours of GPU time in total for this experiment alone. In the other experiments we trained our model for up to 500,000 iterations, which is necessary particularly if data augmentation is applied. Table V shows the comparison of our method with DRDNN. As can be seen, we clearly outperform DRDNN in terms of signal to distortion ratio (SDR) in 3 out of the 4 sources already with our baseline model. Note that this is with a not fully trained model and without any augmentation. As post processing we overlap the segments with an overlap factor of $\gamma = 0.3$, as was done by Li et al. [12]. If we additionally use thresholding, we get even better results. The improvement over DRDNN is especially apparent when we look at the average SDR across sources, which is 2.92 for DRDNN, 3.40 for our baseline and 3.48 for our baseline with thresholding.

To compare against the state of the art in singing voice separation, we choose models that perform best for the vocals source. The results are shown in Table VI. Our baseline model is already competitive with most previous methods. When adding augmentation and/or thresholding, our model clearly outperforms the previous state of the art.

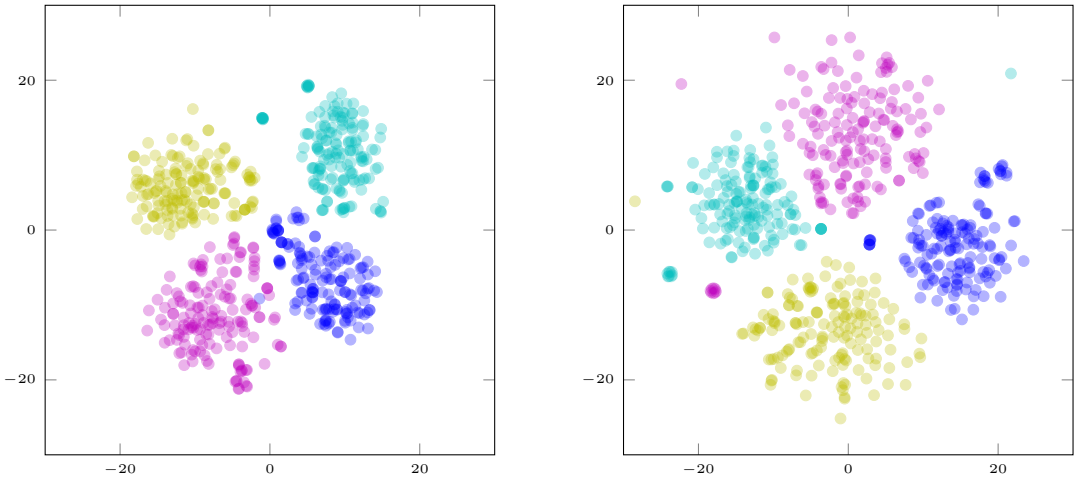
[2]Audio samples of our approach are available here: https://github.com/sveinnpalsson/sourceseparation

Method	Source	SDR	SIR	SAR
RA	Vocals	4.28	10.74	6.00
	Drums	5.00	12.95	6.11
	Bass	2.53	7.14	5.78
	Other	1.12	6.11	3.80
RA+aug	Vocals	**5.22**	11.94	**6.83**
	Drums	5.75	14.12	6.77
	Bass	2.93	7.7	**6.31**
	Other	1.85	7.34	4.07
RA+aug+th	Vocals	5.20	**13.95**	6.40
	Drums	5.62	**14.81**	6.64
	Bass	3.65	8.70	5.44
	Other	3.25	7.38	4.09
RA+aug+th+o	Vocals	**5.22**	12.91	6.64
	Drums	**5.83**	14.13	**7.46**
	Bass	**3.74**	**9.42**	5.82
	Other	**3.29**	**7.69**	**4.42**

TABLE VII: MUSDB18 results for 30s windows using four separate decoders. Values are in [dB]

D. Evaluation on MUSDB18

To ensure direct comparability of our results to future work we provide results on MUSDB18 using both 1s and 30s windows for the evaluation, as proposed in SiSec 2016 and SiSec 2018 respectively. The results are shown in Tables VII and VIII. Clearly, the added amount of training data helps (see Tables II and V). Also, in agreement with the evaluations



(a) Shared decoder.

(b) Specialized decoders.

Fig. 3: t-SNE visualization of sources in the latent space with a shared decoder (a) and specialized decoders (b). Vocals (blue), drums (magenta), bass (yellow) and other (cyan)

Method	Source	SDR	SIR	SAR
RA	Vocals	3.05	9.71	4.12
	Drums	4.23	10.26	4.96
	Bass	3.24	5.85	4.54
	Other	1.71	4.39	2.91
RA+aug	Vocals	**3.52**	**11.79**	4.7
	Drums	4.64	**10.87**	5.52
	Bass	3.59	6.47	5.03
	Other	2.02	**5.68**	3.21
RA+aug+o	Vocals	**3.52**	11.29	**4.74**
	Drums	**4.86**	10.5	**5.98**
	Bass	**3.72**	**6.51**	**5.46**
	Other	**2.08**	5.56	**3.72**

TABLE VIII: MUSDB18 results for 1s windows using four separate decoders. Values are in [dB].

Method	Source	SDR	SIR	SAR
Shared Decoder	Vocals	5.06	11.36	6.69
	Drums	5.53	**14.34**	6.42
	Bass	2.77	7.56	6.13
	Other	1.62	**7.57**	3.83
Separate Decoders	Vocals	**5.22**	**11.94**	**6.83**
	Drums	**5.75**	14.12	**6.77**
	Bass	**2.93**	**7.70**	**6.31**
	Other	**1.85**	7.34	**4.07**

TABLE IX: Test performance of our baseline model trained on MUSDB18 with shared and specialized decoders. Values are in [dB].

presented earlier, data augmentation as well as post processing further increase performance for all sources measured by all metrics. As is apparent from Table VIII, the performance is greatly underestimated when using 1 second windows due to negative outliers. Note that we do not show the results of thresholding in this case, because it could exclude a large number of the evaluation windows and the results would not be directly comparable to other methods. Note that before performing the final experiments on MUSDB18 we found that using separate specialized decoders instead of a single shared decoder yields slightly better results (see also Section V-E. Hence the results on MUSDB18 were achieved using separate decoders.

E. Investigating our Architecture Choice

We base our architecture on the idea of separating sources in the latent representation. However, one might also argue that this is not necessary, as one could also train a separate decoder for each source. Intuitively, a shared decoder forces the encoder and residual network to perform better source separation in the latent space. On the other hand, separate decoders specialize on one type of source (e.g., singing voice), and are thus less prone to wrongly mixing different sources together. We evaluate this design choice by comparing two models trained on MUSDB18: Our baseline model with a shared decoder and a version with specialized decoders. Table IX shows an increase in most performance metrics when using specialized decoders. This means that our results on DSD100 might even be better if we had used separate decoders. To further investigate the source separation in the latent space, we visualize the latent vectors via t-distributed Stochastic Neighbor Embedding (t-SNE) [27], shown in Figure 3. The visualization is created by mapping the test set to the latent space and then computing t-SNE with two components. As expected, the latent vectors of the sources seem to cluster closer together in the case of a shared decoder, which suggests better latent source separation. In the case of specialized decoders the separation is still clear, albeit slightly more spread out. The potential benefit of a (slightly) better latent separation seems to be outweighed by the separate decoders'

specialization on source types. Furthermore, we note that our comparatively simple architecture achieves strong separation in the latent space, suggesting that more complex architectures specifically focusing on latent separation, such as in [12], are not necessary and we should instead focus on improving the reconstruction performance of the decoders, and devising better post processing and data augmentation methods.

VI. Conclusion

We present a simple yet effective method for monaural music source separation based on a ResNet autoencoder. With it we show that state of the art performance is achievable without complex engineering of a specialized architecture. Rather, performance of deep neural networks for monaural music source separation is limited by the size of the data set and whether post processing is applied. We show the effect of several data augmentation methods and post processing tools, which can help to alleviate this problem. Finally, we provide audio samples from our models to show that the increase in SDR indeed translates to better audio quality.

References

[1] J. Peissig and B. Kollmeier, "Directivity of binaural noise reduction in spatial multiple noise-source arrangements for normal and impaired listeners," vol. 101, p. 1660–1670, 1997.

[2] A. Liutkus, F. Stöter, Z. Rafii, D. Kitamura, B. Rivet, N. Ito, N. Ono, and J. Fontecave, "The 2016 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation - 13th International Conference, LVA/ICA 2017, Grenoble, France, February 21-23, 2017, Proceedings*, 2017, pp. 323–332. [Online]. Available: https://doi.org/10.1007/978-3-319-53547-0_31

[3] F. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation - 14th International Conference, LVA/ICA 2018, Guildford, UK, July 2-5, 2018, Proceedings*, 2018, pp. 293–305. [Online]. Available: https://doi.org/10.1007/978-3-319-93764-9_28

[4] Z. Rafii, A. Liutkus, F. Stöter, S. I. Mimilakis, D. FitzGerald, and B. Pardo, "An overview of lead and accompaniment separation in music," *IEEE/ACM Trans. Audio, Speech & Language Processing*, vol. 26, no. 8, pp. 1307–1335, 2018. [Online]. Available: https://doi.org/10.1109/TASLP.2018.2825440

[5] E. Vincent, T. Virtanen, and S. Gannot, *Audio Source Separation and Speech Enhancement*. John Wiley & Sons, 2018.

[6] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, 2014, pp. 1562–1566. [Online]. Available: https://doi.org/10.1109/ICASSP.2014.6853860

[7] ——, "Joint optimization of masks and deep recurrent neural networks for monaural source separation," *IEEE/ACM Trans. Audio, Speech & Language Processing*, vol. 23, no. 12, pp. 2136–2147, 2015. [Online]. Available: https://doi.org/10.1109/TASLP.2015.2468583

[8] M. Miron, J. Janer, and E. Gómez, "Monaural score-informed source separation for classical music using convolutional neural networks," in *ISMIR, 2017*, 2017, pp. 55–62. [Online]. Available: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/51_Paper.pdf

[9] P. Chandna, M. Miron, J. Janer, and E. Gómez, "Monoaural audio source separation using deep convolutional neural networks," in *Latent Variable Analysis and Signal Separation - 13th International Conference, LVA/ICA 2017, Grenoble, France, February 21-23, 2017, Proceedings*, 2017, pp. 258–266. [Online]. Available: https://doi.org/10.1007/978-3-319-53547-0_25

[10] S. I. Mimilakis, K. Drossos, J. F. Santos, G. Schuller, T. Virtanen, and Y. Bengio, "Monaural singing voice separation with skip-filtering connections and recurrent inference of time-frequency mask," *CoRR*, vol. abs/1711.01437, 2017. [Online]. Available: http://arxiv.org/abs/1711.01437

[11] K. Drossos, S. I. Mimilakis, D. Serdyuk, G. Schuller, T. Virtanen, and Y. Bengio, "Mad twinnet: Masker-denoiser architecture with twin networks for monaural sound source separation," *CoRR*, vol. abs/1802.00300, 2018. [Online]. Available: http://arxiv.org/abs/1802.00300

[12] Z. Li, H. Wang, M. Zhao, W. Li, and M. Guo, "Deep representation-decoupling neural networks for monaural music mixture separation," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16733

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[14] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA 2017, New Paltz, NY, USA, October 15-18, 2017*, 2017, pp. 21–25. [Online]. Available: https://doi.org/10.1109/WASPAA.2017.8169987

[15] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," *CoRR*, vol. abs/1805.02410, 2018. [Online]. Available: http://arxiv.org/abs/1805.02410

[16] S. Park, T. Kim, K. Lee, and N. Kwak, "Music source separation using stacked hourglass networks," *CoRR*, vol. abs/1805.08559, 2018. [Online]. Available: http://arxiv.org/abs/1805.08559

[17] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," *CoRR*, vol. abs/1806.03185, 2018. [Online]. Available: http://arxiv.org/abs/1806.03185

[18] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, 2015, pp. 234–241. [Online]. Available: https://doi.org/10.1007/978-3-319-24574-4_28

[19] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[20] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech & Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006. [Online]. Available: https://doi.org/10.1109/TSA.2005.858005

[21] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[23] A. Liutkus and R. Badeau, "Generalized wiener filtering with fractional power spectrograms," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, 2015, pp. 266–270. [Online]. Available: https://doi.org/10.1109/ICASSP.2015.7177973

[24] M. Dolson, "The phase vocoder: A tutorial," vol. 10, p. 14–27, 12 1986.

[25] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, 2015, pp. 121–126. [Online]. Available: http://ismir2015.uma.es/articles/264_Paper.pdf

[26] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. 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, New Orleans, LA, USA, March 5-9, 2017*, 2017, pp. 261–265. [Online]. Available: https://doi.org/10.1109/ICASSP.2017.7952158

[27] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.