DEEP LEARNING ON COMPUTATIONAL ACCELERATORS 236605

PAPER IMPLEMENTATION FINAL REPORT

Monoaural Audio Source Separation Using Deep Convolutional Neural Networks
by P. Chandna, M. Miron, J. Janer, E. Gómez

Submitted by
Gloria Groysman 318372729 Danny Priymak 307003434

Technion IIT, spring 2019

Contents

1 Introduction

Source separation is the separation of a set of source signals from a set of mixed signals, without the aid of information (or with very little information) about the source signals or the mixing process. It is most commonly applied in digital signal processing and involves the analysis of mixtures of signals; the objective is to recover the original component signals from a mixture signal. Audio source separation has drawn the attention of researchers, with approaches varying from using timbre models [2], to those exploiting the repetitive nature of music [3]. While being an interesting problem in itself, the separation of sources from a mixture can serve as a intermediary step for other tasks such as automatic speech recognition and fundamental frequency estimation. Some applications, such as speech enhancement for cochlear implant users, require low-latency processing. The original paper we chose to focus on, by Chandna et al. [1] has tackled the audio signal source separation case, and has introduced a low-latency monaural source separation framework using a convolutional neural network (CNN). The authors used a CNN to estimate time-frequency soft masks, generated per-instrument, which in turn are applied to musical tracks in order to isolate the corresponding instrument's estimated, isolated audio. The training, as well as an evaluation of the neural network's performance was performed over a dataset containing fully mixed audio music tracks of four fixed instruments: vocals, drums, and bass, as well as a fourth instrument category representing all other remaining instruments, which vary from track to track. The authors' proposed architecture has been compared to a Multilayer Perceptron (MLP) architecture, and achieved on-par results in conjunction with a significant improvement in processing time. Our work intends to implement this architecture and provide possible enhancements.

2 Related Work

Several approaches have been proposed for the solution of this problem but development is currently still very much in progress. A more successful classical approach that does not involve machine learning concepts is non-negative matrix factorization (NMF). Additional popular approaches are principal components analysis and independent component analysis, which work well when there are no delays or echoes present; that is, the problem is simplified a great deal.

Approaches directly using deep neural networks for separation have been proposed as well. Nugraha et al. [4] adapt deep neural networks for multichannel source separation, using both phase and magnitude information. As for monaural source separation, Huang et al. [5] propose a method that utilizes a deep neural network, taking a single frame of the magnitude spectrogram of a mixture as an input feature to learn single-frame timbre features for each source. Temporal evolution is then modeled using a recurrent layer. Uhlich et al. [6] propose another method which takes multiple frames of the magnitude spectrogram of a mixture as input and consists of only fully connected layers. This method models timbre features across multiple time frames. While these approaches work well, they do not exploit completely local time-frequency features. Instead, they rely on global features across the entire frequency spectrum, over a longer period of time.

3 Method

Convolutional neural networks (CNNs), take advantage of small scale features present in data. CNNs require less memory and resources than regular fully connected neural networks, allowing for a faster, more efficient model. CNNs have proved to be successful in image processing for tasks such as image super-resolution and semantic segmentation of images. In the image processing field, CNNs take as input a two-dimensional vector of pixel intensities across the spatial dimension and exploit the local spatial correlation among input neurons to learn localized features. A similar two-dimensional representation is used in our model for audio mixtures, using the Short-Time Fourier Transform (STFT), which has frequency and time dimensions. Unlike 2D images, the STFT does not have symmetry across both axes, but a local symmetry can be found along each single axes. Therefore, the filters used in CNNs need to be adapted to the STFT representation of audio.

Figure 1 shows the block diagram for the proposed source separation framework. The STFT is computed on a segment of time context T of the mixture audio. The resulting magnitude spectrogram is then passed through the network, which outputs an estimate for each of the separated sources. The estimate is used to compute time-frequency soft masks, which are applied to the magnitude spectrogram of the mixture to compute final magnitude estimates for the separated sources. These estimates, along with the phase of the mixture, are used to obtain the audio signals corresponding to the separated sources.

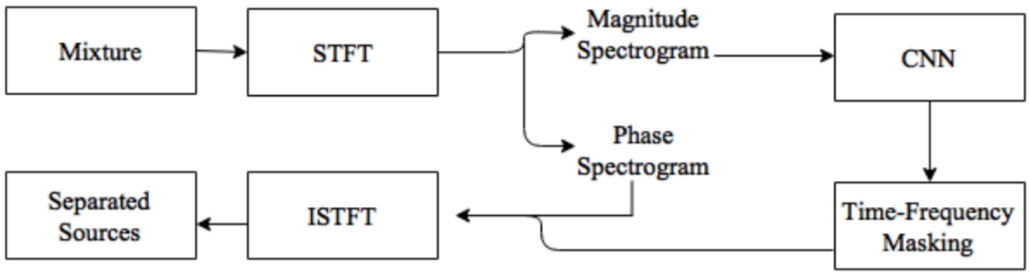

Figure 1: Data Flow

3.1 Architecture

The authors use a CNN which functions as a variation of an auto encoder architecture. The network is able to learn an end-to-end model for the separated sources by finding a compressed representation for the training data. The model proposed is shown in Figure 2.

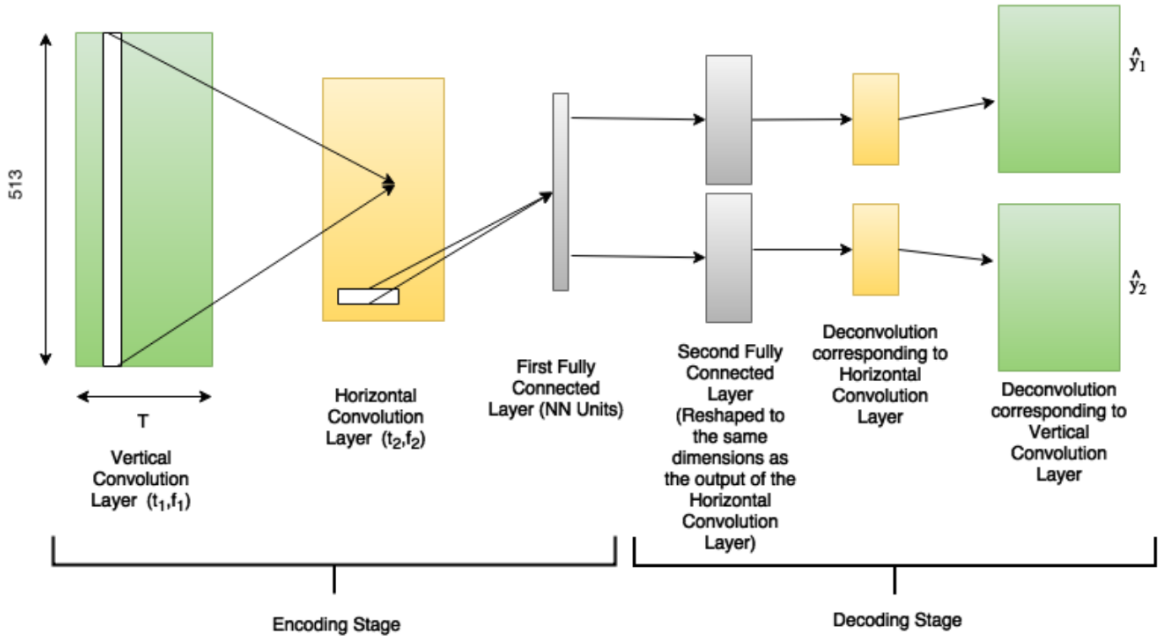

Figure 2: Network Architecture for Source Separation, Using Vertical And Horizontal Convolutions, Showing The Encoding And Decoding Stages

It uses a CNN with two stages: a convolution or encoding stage and the inverse operation, the deconvolution or decoding stage. The stages are explained in further details below.

Encoding Stage. This part of the network consists of two convolution layers and a fully connected dense layer, which acts as a bottleneck to compress information.

1. **Vertical convolution layer.** This convolution layer has a shape of (t_1, f_1) , spanning across a t_1 time frame and taking into account f_1 frequency bins. This layer tries to capture local timbre information, allowing the model to learn timbre features, similar to the approach used in NMF algorithms for source separation. These features are shared among the sources to be separated, contrary to the NMF approach, where specific basis and activation gains are derived for each source. Therefore, the timbre features learned by this layer need to be robust enough to separate the required source across songs of different genres, where the type of instruments and singers might vary. N_1 filters were used in this layer.

2. **Horizontal convolution layer.** This layer models temporal evolution for different instruments from the features learned in the vertical convolution layer. This is particularly useful for modeling time-frequency characteristics of the different instruments present in the sources to be separated. The filter shape of this layer is (t_2, f_2) and N_2 filters were used.

3. **Fully connected layer.** The output of the horizontal convolution layer is connected to a fully connected Rectified Linear Unit (ReLU) layer which acts as a bottleneck, achieving dimensional reduction. This layer consists of a non-linear combination of the

features learned from the previous layers, with a ReLU non-linearity. The layer is chosen to have fewer elements to reduce the total number of network parameters and to ensure that the network is able to produce a robust representation of the input data. The number of nodes in this layer is represented as NN.

Decoding Stage. The output of the first fully connected layer is passed to another fully connected layer, with a ReLU non-linearity and the same size as the output of the second convolution layer. Thereafter, this layer is reshaped to the same dimensions as the horizontal convolution layer and passed through successive deconvolution layers, the inverse operations to the convolution stage.

3.2 Time-Frequency Masking

The authors wished integrate the computation of a softmask for each of the sources into the network. From the output of the network $\hat{y}_n(f)$, a soft mask $m_n(f)$ can be computed as follows

$$m_n(f) = \frac{|\hat{y}_n(f)|}{\sum_{n=1}^{N} |\hat{y}_n(f)|}$$

where $\hat{y}_n(f)$ represents the output of the network for the n-th source and N is the total number of sources to be estimated. The estimated mask is then applied to the input mixture signal to estimate the sources $\widetilde{y}_n$

$$\widetilde{y}_n(f) = m_n(f)\, x(f)$$

where $x(f)$ is the spectrogram of the input mixture signal.

3.3 Loss Function

Initially, the neural network was intended to be trained to optimize parameters using a Stochastic Gradient Descent (SGD) in order to minimize the squared error between the estimate $\widetilde{y}_n$ and the original source y_n. Formally

$$L_{\text{sq}} = \sum_{i=1}^{N} \|\widetilde{y}_n - y_n\|^2$$

And as the authors mentioned in the original paper, the learning objective has later been changed to

$$L_{\text{total}} = L_{\text{sq'}} - \alpha \cdot L_{\text{diff}} - \beta \cdot L_{\text{other}} - \beta_{\text{vocals}} \cdot L_{\text{othervocals}}$$

where $\alpha, \beta, \beta_{\text{vocals}}$ are hyper parameters, and (note the sum indices)

$$L_{\text{sq'}} = \sum_{i=1}^{N-1} \|\widetilde{y}_n = y_n\|^2 \qquad L_{\text{diff}} = \sum_{n=1}^{N-1} \left\|\widetilde{y}_n - \widetilde{y}_{\widetilde{n}\neq n}\right\|^2$$

$$L_{\text{othervocals}} = \|\widetilde{y}_1 - y_N\|^2 \qquad L_{\text{other}} = \sum_{n=2}^{N-1} \|\widetilde{y}_n - y_N\|^2$$

Lastly, y_1, y_N represent the sources corresponding to vocals and other instruments, respectively.

4 Training

4.1 Dataset

The authors used the **Demixing Secrets Dataset 100** (DSD100) for training and testing. This dataset consists of 100 professionally produced full track songs and is designed to evaluate signal source separation methods from music recordings. The dataset contains separate tracks for drums, bass, vocals and other instruments for each song in the set, present as stereo wav files with a sample rate of 44.1 KHz. The four source tracks are mixed using a professional Digital Audio Workstation to create the mixture track for each song. The dataset is divided into a dev set, used for training the network and a test set, which is used for testing the network. Both of these subsets consist of 50 songs each.

4.2 Training Procedure

During the training phase, the input mixture and the individual sources comprising the mixture were split into 20 seconds segments, and the short time Fourier transform (STFT) for each of these segments was computed. A Hanning window of length 1024 samples was used, which, at a sampling rate of 44.1 KHz corresponds to 23 milliseconds (ms), and a hopsize of 256 samples (5.8 ms), leading to an overlap of 75% across frames. The frames generated from this procedure were grouped into batches of T frames, representing the maximum time context that the network tries to model. The batches were fed to the model for training, with 30 batches being fed at each round. Thus, each batch consists of T frames of 513 frequency bins. A complete pass over the entire set is considered as one training epoch and the network is trained for 30 epochs.

5 Evaluation

For evaluation, the following measures were used: Source to Distortion Ratio (SDR), Source to Interference Ratio (SIR), Source to Artifacts Ratio (SAR), and Image to Spatial distortion Ratio (ISR). These measures are averaged for overlapping 30 second frames of each song in both the training and the test set.

6 Our Experiments and Results

Tables 1 - 4 represent the results achieved by our experiments in which we tweaked the original network's parameters. The first experiment investigates our results for the same parameters as the original authors for us to have a baseline to build upon. Upon inspecting its first results, we noticed signs of overfitting. As a result, all following experiments were changed to use a ratio of 80%/20% of training/test tracks, respectively, due to the need for a larger dataset to train on and a lack of need for a large test set. The next 3 experiments investigated batch

size changes, to test whether a smaller batch size will yield less sample noise than a larger batch size. The 5th experiment investigated different network layer sizes in conjunction with maxpooling in order to decrease the total number of learned parameters. The 6th experiment investigated dropout to the first fully connected layer in order to perform regularization and not be dependent of specific weights, and by that to, hopefully, prevent overfitting. The 7h and 8th experiments investigated different learning rate tweaks to hopefully reach the global minimum more easily. All experiments included a milestone mechanism, which decreased the learning rate by a factor of 10 at every milestone. Additionally, all experiments were performed using an early stopping criterion, with a patience epoch size of 7, since training up to the full number of 200 epochs did not yield any result improvement and caused overfitting. In practice, the experiments stopped (due to the early stopping criterion) before reaching the first milestone. A small sample of our results as audio files can be found in the project as well. Lastly, an instructions text file is also provided to guide others who would like to reproduce our experiments. The instructions show which Python scripts should be run to prepare the dataset, train the model and test it.

Experiment 1

Our configuration (Same as authors' original configuration) (Train Set, Test + Validation Set, Batch Size) $= (50\%, 50\%, 30)$ $(t_1, f_1, t_2, f_2, N_1, N_2, NN) = (1, 513, 12, 1, 50, 30, 128)$ $(\alpha, \beta, \beta_{\text{vocals}}) = (0.001, 0.01, 0.03)$ $(\text{Epochs}, \text{Learning Rate}, \text{Milestones}) = \left(200, 1 \times 10^{-5}, [60, 120]\right)$ $(\text{Training Loss}, \text{Validation Loss}) = (18473.046, 97306.92)$			Authors' Results
SDR [dB]	Bass	-22.17 ± 2.40	0.9 ± 2.7
	Drums	-25.37 ± 1.62	2.4 ± 2.0
	Others	-22.25 ± 2.01	0.8 ± 1.5
	Vocals	-22.73 ± 2.51	1.3 ± 2.4
SIR [dB]	Bass	-2.10 ± 2.72	4.6 ± 4.4
	Drums	-5.10 ± 1.67	9.1 ± 4.3
	Others	-2.39 ± 2.05	3.8 ± 4.0
	Vocals	-3.30 ± 2.17	7.2 ± 3.6
SAR [dB]	Bass	-17.74 ± 1.54	6.9 ± 2.3
	Drums	-18.99 ± 1.23	7.0 ± 2.8
	Others	-17.72 ± 1.50	2.8 ± 2.4
	Vocals	-17.59 ± 1.69	5.3 ± 2.9

Table 1: Experiment 1 investigating a 50%/50% training and test sets distribution of the dataset, as the original authors.

Experiment 2

Our configuration (Train Set, Validation Set, Test Set, Batch Size) $= (80\%, 10\%, 10\%, 30)$ $(t_1, f_1, t_2, f_2, N_1, N_2, NN) = (1, 513, 12, 1, 50, 30, 128)$ $(\alpha, \beta, \beta_{\text{vocals}}) = (0.001, 0.01, 0.03)$ (Epochs, Learning Rate, Milestones) $= \left(200, 1 \times 10^{-3}, [60, 120]\right)$ (Training Loss, Validation Loss) $= (20538.27, 111510.24)$			Authors' Results
SDR [dB]	Bass	-21.270 ± 3.08	0.9 ± 2.7
	Drums	-25.027 ± 1.82	2.4 ± 2.0
	Others	-22.087 ± 1.78	0.8 ± 1.5
	Vocals	-21.70 ± 2.21	1.3 ± 2.4
SIR [dB]	Bass	-1.45 ± 3.23	4.6 ± 4.4
	Drums	-4.44 ± 2.24	9.1 ± 4.3
	Others	-2.57 ± 1.64	3.8 ± 4.0
	Vocals	-1.41 ± 2.05	7.2 ± 3.6
SAR [dB]	Bass	-17.14 ± 1.49	6.9 ± 2.3
	Drums	-19.09 ± 0.68	7.0 ± 2.8
	Others	-17.47 ± 1.31	2.8 ± 2.4
	Vocals	-17.77 ± 1.35	5.3 ± 2.9

Experiment 3

Our configuration (Train Set, Validation Set, Test Set, Batch Size) $= (80\%, 10\%, 10\%, 5)$ $(t_1, f_1, t_2, f_2, N_1, N_2, NN) = (1, 513, 12, 1, 50, 30, 128)$ $(\alpha, \beta, \beta_{\text{vocals}}) = (0.001, 0.01, 0.03)$ (Epochs, Learning Rate, Milestones) $= \left(200, 1 \times 10^{-3}, [60, 120]\right)$ (Training Loss, Validation Loss) $= (21876.123, 37201.086)$			Authors' Results
SDR [dB]	Bass	$-21.60 \pm 2.97.$	0.9 ± 2.7
	Drums	-24.74 ± 1.39	2.4 ± 2.0
	Others	-22.76 ± 1.68	0.8 ± 1.5
	Vocals	-23.58 ± 2.086	1.3 ± 2.4
SIR [dB]	Bass	-1.41 ± 2.72	4.6 ± 4.4
	Drums	-4.81 ± 2.087	9.1 ± 4.3
	Others	-3.476 ± 1.81	3.8 ± 4.0
	Vocals	-4.32 ± 2.02	7.2 ± 3.6
SAR [dB]	Bass	-17.584 ± 1.65	6.9 ± 2.3
	Drums	-18.54 ± 1.00	7.0 ± 2.8
	Others	-17.54 ± 1.08	2.8 ± 2.4
	Vocals	-17.74 ± 1.33	5.3 ± 2.9

Experiment 4

Our configuration (Train Set, Validation Set, Test Set, Batch Size) $= (80\%, 10\%, 10\%, 128)$ $(t_1, f_1, t_2, f_2, N_1, N_2, NN) = (1, 513, 12, 1, 50, 30, 128)$ $(\alpha, \beta, \beta_{\text{vocals}}) = (0.001, 0.01, 0.03)$ $(\text{Epochs}, \text{Learning Rate}, \text{Milestones}) = \left(200, 1 \times 10^{-3}, [60, 120]\right)$ $(\text{Training Loss}, \text{Validation Loss}) = (14560.074, 144205.705)$			Authors' Results
SDR [dB]	Bass	$-21.61 \pm 2.53.$	0.9 ± 2.7
	Drums	-24.37 ± 1.23	2.4 ± 2.0
	Others	-23.22 ± 1.53	0.8 ± 1.5
	Vocals	-23.43 ± 2.08	1.3 ± 2.4
SIR [dB]	Bass	-1.59 ± 2.77	4.6 ± 4.4
	Drums	-4.44 ± 1.83	9.1 ± 4.3
	Others	-2.97 ± 1.22	3.8 ± 4.0
	Vocals	-4.32 ± 2.02	7.2 ± 3.6
SAR [dB]	Bass	-17.480 ± 1.29	6.9 ± 2.3
	Drums	-18.46 ± 0.87	7.0 ± 2.8
	Others	-18.39 ± 1.10	2.8 ± 2.4
	Vocals	-18.07 ± 1.37	5.3 ± 2.9

Table 2: Experiments 2 - 4 investigating batch size changes.

Experiment 5

Our configuration (Train Set, Validation Set, Test Set, Batch Size) $= (80\%, 10\%, 10\%, 30)$ $(t_1, f_1, t_2, f_2, N_1, N_2, NN) = (1, 513/3, 12, 1, 50, 30/5, 128)$ $(\alpha, \beta, \beta_{\text{vocals}}) = (0.001, 0.01, 0.03)$ $(\text{Epochs}, \text{Learning Rate}, \text{Milestones}) = \left(200, 1 \times 10^{-3}, [60, 120]\right)$ Convolutional layer kernel size changed to $(513/3, 1)$ Added maxpool with kernel size $(2 \cdot (513/3), 1)$ Added unpool $(\text{Training Loss}, \text{Validation Loss}) = (17156.322, 114269.075)$			Authors' Results
SDR [dB]	Bass	$-22.32 \pm 2.26.$	0.9 ± 2.7
	Drums	-24.92 ± 0.92	2.4 ± 2.0
	Others	-23.01 ± 1.25	0.8 ± 1.5
	Vocals	-23.16 ± 1.88	1.3 ± 2.4
SIR [dB]	Bass	-2.25 ± 1.68	4.6 ± 4.4
	Drums	-5.68 ± 1.58	9.1 ± 4.3
	Others	-3.44 ± 1.59	3.8 ± 4.0
	Vocals	-3.66 ± 1.94	7.2 ± 3.6
SAR [dB]	Bass	-17.91 ± 1.44	6.9 ± 2.3
	Drums	-18.09 ± 0.92	7.0 ± 2.8
	Others	-17.83 ± 0.94	2.8 ± 2.4
	Vocals	-17.80 ± 1.13	5.3 ± 2.9

Experiment 6

Our configuration			Authors' Results
(Train Set, Validation Set, Test Set, Batch Size) $= (80\%, 10\%, 10\%, 30)$ $(t_1, f_1, t_2, f_2, N_1, N_2, NN) = (1, 513, 12, 1, 50, 30, 128)$ $(\alpha, \beta, \beta_{\text{vocals}}) = (0.001, 0.01, 0.03)$ $(\text{Epochs}, \text{Learning Rate}, \text{Milestones}) = \left(200, 1 \times 10^{-3}, [60, 120]\right)$ Same as experiment 5, but added dropout to fully connected layer fc0 $(\text{Training Loss}, \text{Validation Loss}) = (22073.190, 117374.315)$			
SDR [dB]	Bass	$-21.50 \pm 2.80.$	0.9 ± 2.7
	Drums	-25.07 ± 1.63	2.4 ± 2.0
	Others	-22.71 ± 1.84	0.8 ± 1.5
	Vocals	-23.75 ± 2.10	1.3 ± 2.4
SIR [dB]	Bass	-1.32 ± 2.72	4.6 ± 4.4
	Drums	-5.10 ± 2.51	9.1 ± 4.3
	Others	-3.17 ± 1.78	3.8 ± 4.0
	Vocals	-4.52 ± 2.05	7.2 ± 3.6
SAR [dB]	Bass	-17.53 ± 1.54	6.9 ± 2.3
	Drums	-18.61 ± 1.14	7.0 ± 2.8
	Others	-17.69 ± 1.17	2.8 ± 2.4
	Vocals	-17.76 ± 1.20	5.3 ± 2.9

Table 3: Experiments 5 and 6 investigating different network layer sizes and adding dropout to the fully connected fc0 layer.

Experiment 7

Our configuration			Authors' Results
(Train Set, Validation Set, Test Set, Batch Size) $= (80\%, 10\%, 10\%, 30)$ $(t_1, f_1, t_2, f_2, N_1, N_2, NN) = (1, 513, 12, 1, 50, 30, 128)$ $(\alpha, \beta, \beta_{\text{vocals}}) = (0.001, 0.01, 0.03)$ $(\text{Epochs}, \text{Learning Rate}, \text{Milestones}) = \left(200, 1 \times 10^{-5}, [60, 120]\right)$ $(\text{Training Loss}, \text{Validation Loss}) = (21492.434, 115899.219)$			
SDR [dB]	Bass	-21.82 ± 2.37	0.9 ± 2.7
	Drums	-24.99 ± 1.67	2.4 ± 2.0
	Others	-22.21 ± 1.19	0.8 ± 1.5
	Vocals	-23.08 ± 1.90	1.3 ± 2.4
SIR [dB]	Bass	-1.60 ± 2.57	4.6 ± 4.4
	Drums	-5.62 ± 2.56	9.1 ± 4.3
	Others	-2.97 ± 1.62	3.8 ± 4.0
	Vocals	-4.11 ± 1.94	7.2 ± 3.6
SAR [dB]	Bass	-17.72 ± 1.10	6.9 ± 2.3
	Drums	-18.12 ± 1.06	7.0 ± 2.8
	Others	-17.34 ± 1.26	2.8 ± 2.4
	Vocals	-17.39 ± 1.35	5.3 ± 2.9

Experiment 8

Our configuration			Authors' Results
(Train Set, Validation Set, Test Set, Batch Size) $= (80\%, 10\%, 10\%, 30)$ $(t_1, f_1, t_2, f_2, N_1, N_2, NN) = (1, 513, 12, 1, 50, 30, 128)$ $(\alpha, \beta, \beta_{\text{vocals}}) = (0.001, 0.01, 0.03)$ (Epochs, Learning Rate, Milestones) $= \left(200, 1 \times 10^{-5}, [60, 120]\right)$ Changed learning rate to 1×10^{-6} after 10 epochs followed by 1×10^{-7} after 50 epochs (Training Loss, Validation Loss) $= (29001.09, 127572.20)$			
SDR [dB]	Bass	-22.45 ± 1.89	0.9 ± 2.7
	Drums	-25.11 ± 1.41	2.4 ± 2.0
	Others	-22.63 ± 1.32	0.8 ± 1.5
	Vocals	-23.49 ± 1.94	1.3 ± 2.4
SIR [dB]	Bass	-2.57 ± 2.01	4.6 ± 4.4
	Drums	-6.31 ± 2.28	9.1 ± 4.3
	Others	-3.48 ± 1.69	3.8 ± 4.0
	Vocals	-4.55 ± 1.99	7.2 ± 3.6
SAR [dB]	Bass	-17.82 ± 1.00	6.9 ± 2.3
	Drums	-17.73 ± 1.03	7.0 ± 2.8
	Others	-17.41 ± 1.16	2.8 ± 2.4
	Vocals	-17.49 ± 1.21	5.3 ± 2.9

Table 4: Experiments 7 and 8 investigating varying learning rates.

References

[1] P. Chandna, M. Miron, J. Janer, E. G´omez. Monoaural audio source separation using deep convolutional neural networks. In 13th International Conference on Latent Variable Analysis and Signal Separation (LVAICA2017), 02/2017 2017

[2] Durrieu, J., Ozerov, A., and F´evotte, C. (2009). Main instrument separation from stereophonic audio signals using a source/filter model. 17th European Signal Processing Conference.

[3] Rafii, Z., Liutkus, A., and Pardo, B. (2014). REPET for Background/Foreground Separation in Audio, pages 395–411. Springer Berlin Heidelberg, Berlin, Heidelberg.

[4] Nugraha, A. A., Liutkus, A., and Vincent, E. (2016). Multichannel audio source separation with deep neural networks. Technical report.

[5] Huang, P.-S., Kim, M., Hasegawa-Johnson, M., and Smaragdis, P. (2014). Deep Learning for Monaural Speech Separation. Acoustics, Speech and Signal Processing (ICASSP), pages 1562–1566.

[6] Uhlich, S., Giron, F., and Mitsufuji, Y. (2015). Deep neural network basedinstrument extraction from music. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2135–2139. IEEE.