Monoaural Audio Source Separation Using Deep Convolutional Neural Networks

Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez

Music Technology Group, Universitat Pompeu Fabra, Barcelona
`pritish.chandna@upf.edu`
`marius.miron,jordi.janer,emilia.gomez@upf.edu`
`http://mtg.upf.edu`

Abstract. In this paper we introduce a low-latency monaural source separation framework using a Convolutional Neural Network (CNN). We use a CNN to estimate time-frequency soft masks which are applied for source separation. We evaluate the performance of the neural network on a database comprising of musical mixtures of three instruments: voice, drums, bass as well as other instruments which vary from song to song. The proposed architecture is compared to a Multilayer Perceptron (MLP), achieving on-par results and a significant improvement in processing time. The algorithm was submitted to source separation evaluation campaigns to test efficiency, and achieved competitive results.

Key words: convolutional autoencoder, music source separation, deep learning, convolutional neural networks, low-latency

1 Introduction

Monoaural audio source separation has drawn the attention of many researchers in the past few years, with approaches varying from using timbre models such as those proposed by [4], to those exploiting the repetitive nature of music such as [13]. 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, [9] and fundamental frequency estimation, [5]. Some applications, such as speech enhancement for cochlear implant users, [9] [7], require low-latency processing, which we will focus on in this paper.

Techniques using Non-Negative Matrix Factorization (NMF) have historically been the most prominent in this field, as seen in [4]. While effective, these approaches have a high processing time and are difficult to adapt for real-time applications.

Approaches directly using deep neural networks for separation have been proposed recently. A deep architecture for estimating Ideal Binary Masks (IBMs) to separate speech signals from a noisy mixture was proposed by [18]. Nugraha et al. [12] adapt deep neural networks for multichannel source separation, using both phase and magnitude information. With respect to monaural source separation, Huang et al. [8] propose a method using deep neural networks, which takes 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. [16] 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. Convolutional neural networks (CNNs), as seen in [10], 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 recently been used by [15] for extracting vocals from a musical mixture.

CNNs have proved to be successful in image processing for tasks such as image super-resolution [3] and semantic segmentation of images as proposed by [11]. 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 axis, 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. To this end, a network architecture is proposed in Section 2. In Section 3 we evaluate the proposed model on the DSD100 dataset for the separation of four sources from a mix and compare the results with a Multilayer Perceptron architecture.

2 Proposed framework

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.

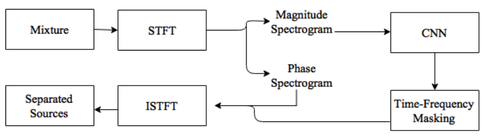

Fig. 1. Data Flow

2.1 Model Architecture

State-of-the-art deep learning frameworks model source separation as a regression problem, where the network yields full resolution output for all the sources. This elicits a high numbers of parameters, which increase the processing time of the network. We take advantage of the parameter reduction property of a CNN architecture to alleviate this problem. In order to keep the multi-scale reasoning used in classification problems, we use a CNN which functions as a variation of an autoencoder architecture, as used by [14]. 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 in this paper is shown in Figure 2. It uses a CNN with two stages, a convolution or encoding stage and the inverse operation, the deconvolution or decoding stage. We use vertical and horizontal convolutions, which have been successfully used in automatic speech recognition [6][1].

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 the shape (t_1, f_1), spanning across 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 [14]. 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 parameters of the network 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

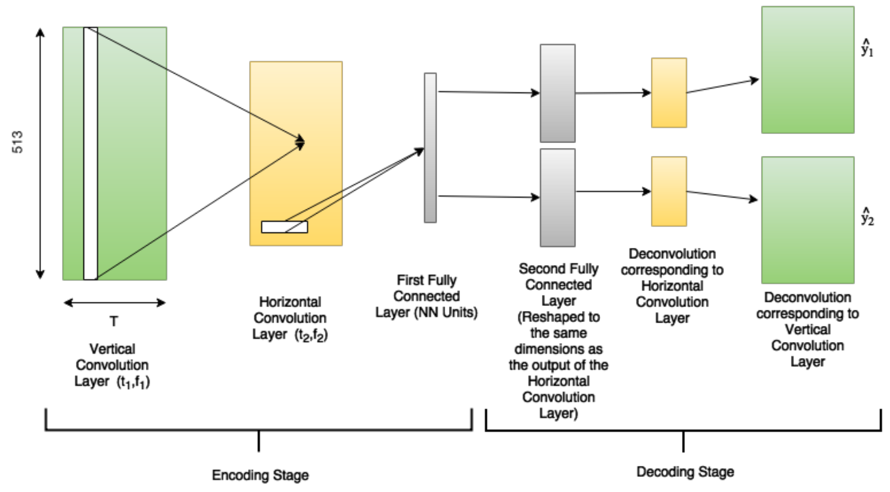
513
T
Vertical Convolution Layer (t1,f1)
Horizontal Convolution Layer (t2,f2)
First Fully Connected Layer (NN Units)
Second Fully Connected Layer (Reshaped to the same dimensions as the output of the Horizontal Convolution Layer)
Deconvolution corresponding to Horizontal Convolution Layer
Deconvolution corresponding to Vertical Convolution Layer
ŷ1
ŷ2
Encoding Stage
Decoding Stage

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

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. This approach is similar to the one proposed by [11] and is repeated to compute estimates, $\hat{y}_n$, for each of the sources, y_n.

2.2 Time-frequency masking

As advocated in [18][8], it is desirable to integrate the computation of a soft mask for each of the sources into the network. From the output of the network $\hat{y}_n(f)$, we can compute a soft mask, $m_n(f)$ as follows:

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

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 $\tilde{y}_n$.

$$\tilde{y}_n(f) = m_n(f)x(f) \tag{2}$$

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

2.3 Parameter learning

The neural network is trained to optimize parameters using a Stochastic Gradient Descent with AdaDelta algorithm, as proposed by [19], in order to minimize the squared error between the estimate and the original source y_n.

$$L_{sq} = \sum_{i=1}^{N} \|\tilde{y}_n - y_n\|^2 \tag{3}$$

3 Evaluation

3.1 Dataset

We consider the *Demixing Secrets Dataset 100* (DSD100) for training and testing our proposed architecture. This dataset consist of 100 professionally produced full track songs from *The Mixing Secrets Free Multitrack Download Library* 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 sets consist of 50 songs each.

3.2 Adjustments To Learning Objective

After some initial experimentation, we observed that an additional loss term, L_{diff}, representing the difference between the estimated sources, as used by [8], improved the performance of the system. In addition, we observed that, while voice, bass and drums were consistently present across songs, the other instruments varied a lot. Thus, the network was not able to efficiently learn a representation for this category as it tries to learn a general timbre class instead of particularities of the different instruments to be separated. We overcame this by modifying L_{sq} to $L_{sq'}$ and incorporating an additional loss term, L_{other}. $L_{sq'}$ excludes the source corresponding to other instruments, while L_{other} encourages differences between sources such as 'vocals' and 'other', 'bass' and 'other', and 'drums' and 'other'.

Also, we noted that the 'other' source comprised of harmonic instruments such as guitars and synths, which were similar to the 'vocals' source. To emphasize the difference between these two sources in the separation stage, a $L_{othervocals}$ loss element, which represents the difference between the estimated vocals and the other stem, was introduced.

$$L_{sq'} = \sum_{i=1}^{N-1} \|\tilde{y}_n - y_n\|^2 \tag{4}$$

$$L_{diff} = \sum_{n=1}^{N-1} \|\tilde{y}_n - \tilde{y}_{\hat{n} \neq n}\|^2 \qquad L_{othervocals} = \|\tilde{y}_1 - y_N\|^2 \qquad L_{other} = \sum_{n=2}^{N-1} \|\tilde{y}_n - y_N\|^2 \tag{5}$$

The total cost is then written as:

$$L_{total} = L_{sq'} - \alpha L_{diff} - \beta L_{other} - \beta_{vocals} L_{othervocals} \tag{6}$$

y_1 represents the source corresponding to vocals and y_N represents that corresponding to other instruments.

3.3 Evaluation setup

During the training phase, the input mixture and the individual sources comprising the mixture were split into 20 seconds segments, and the STFT for each of these segments was computed. We used a Hanning window of length 1024 samples, 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. Batches were also generated using a 50% overlap to generate more data for training. These batches were shuffled to avoid over-fitting and 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, an experimentally determined variable. Lasagne, a framework for neural networks built on top of Theano[1], was used for data flow and network training on a computer with GeForce GTX TITAN X GPU, Intel Core i7-5820K 3.3GHz 6-Core Processor, X99 gaming 5 x99 ATX DDR44 motherboard.

For evaluation, the measures proposed by [17] were used. These include: *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 development and the test set.

3.4 Experiments

The number of parameters of the network is directly proportional to the processing time required by the network. Since our aim was to design a low-latency source separation algorithm, we tried to minimize the parameters of the network, by adjusting the variables, Time context T in frames, Filter shapes (t_1, f_1) and (t_2, f_2), the number of filters, N_1 and N_2 and the number of nodes in the bottleneck, NN, while not compromising on performance. These variables were determined to be 25 (290 ms), $(1, 513)$, $(12, 1)$, 50, 30 and 128 respectively. For more

[1] `http://lasagne.readthedocs.io/en/latest/`Lasagne and `http://deeplearning.net/software/theano/`Theano

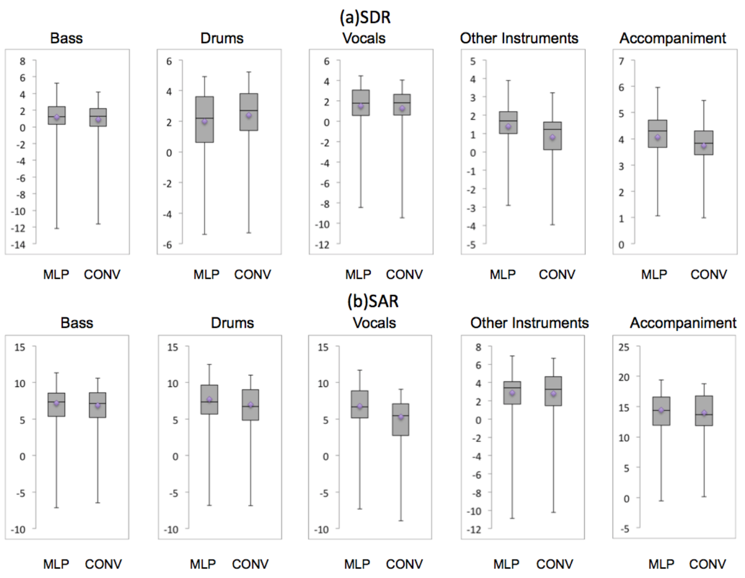

Fig. 3. Comparison between the output of the Multilayer Perceptron (MLP) and the Convolutional Neural Network (CONV) in terms of (a) SDR and (b) SAR

details on these experiments, please refer to [2]. The parameters α, β and β_{vocals} were also experimentally determined to be 0.001, 0.01 and 0.03 respectively. [2]

The evaluation of the CNN model and an MLP with similar training criterion is shown in Table 1, for each of the four aforementioned sources plus the accompaniment (Acc.), which refers to the entire mix minus the vocals. Moreover, in order to asses low-latency capabilities, the total number of parameters to be optimized and the processing time for a batch of T time frames for each model are also reported. The processing time reported was calculated on the CPU,without the use of the GPU.

Model Details	Measure	Bass	Drums	Vocals	Others	Acc.
Multilayer Perceptron (MLP)	SDR	1.2±2.7	2.0±2.1	1.5±2.7	1.4±1.4	4.1±0.9
	SIR	3.7±4.2	6.5±4.1	6.6±3.7	4.7±4.2	15.0±3.7
Processing Time = 654.3 **ms**	SAR	7.2±2.3	7.7±2.6	6.8±2.5	2.9±2.2	14.4±3.0
6617704+$N \times$ 1654426 params.	ISR	11.4±3.8	8.5±2.4	7.8±3.0	3.7±1.5	6.2±1.3
Convolution With Horizontal	SDR	0.9±2.7	2.4±2.0	1.3±2.4	0.8±1.5	3.7±0.8
And Vertical Filters (CONV)	SIR	4.6±4.4	9.1±4.3	7.2±3.6	3.8±4.0	14.7±3.5
Processing Time = 160.8 **ms**	SAR	6.9±2.3	7.0±2.8	5.3±2.9	2.8±2.4	14.0±3.4
97698+$N \times$ 54181 params.	ISR	11.5±3.4	8.5±2.2	7.3±3.0	4.4±1.7	6.1±1.3

Table 1. Evaluation Results. Values in the table are presented in Decibels: Mean ± Standard Deviation. N represents the number of sources to be separated.

Table 1 and Figure 3 show that the performance of the MLP and CNN architectures was similar. However, a significant increase in the number of parameters, up to $26x$, involved in the network was observed when using an MLP architecture. The processing time required by the MLP on the computer system used was $4x$ higher than the processing time required for the CNN. For an input time from of 290 ms, the CNN took just 161 ms to process, whereas the MLP took an average of 654 ms. This shows that for low-latency requirements, it is preferable to use convolution networks with horizontal and vertical layers over a simple MLP architecture.

Comparison with state-of-the-art The algorithm was submitted to the MIREX2016 singing voice separation task and achieved results on-par with the best algorithms in the challenge, both in terms of runtime and evaluation metrics. These results can be found at `http://www.music-ir.org/mirex/wiki/2016:Singing_Voice_Separation_Results`. The framework was also submitted to SiSEC 2016, for comparison with other state-of-the-art algorithms for source separation. The evaluation results for this campaign can be found at `https://sisec.inria.fr/`.

Sound examples of applying the model to real-world mp3 songs can be found at `https://www.youtube.com/watch?v=71WwHyNaDfE`, demonstrating the robustness of the model. Source code for the framework can be found on GitHub at `https://github.com/MTG/DeepConvSep`.

4 Conclusions And Future Work

We designed a low-latency monoaural audio source separation algorithm using a deep convolutional neural network. It was noted that the use of convolutional filters specifically designed for audio data allowed a significant gain in processing time over a simple multilayer perceptron. Dimensional reduction in the fully connected layer allows the model the learn a more compact representation of the input data from which the sources can be separated. Contrary to other approaches, which try to model both the target instrument and other background instruments, the presented algorithm solely models the target sources while the stems of the other instruments are used primarily to increase their dissimilarity with the targets.

We plan to explore the potential use of the algorithm for low-latency applications such as remixing for cochlear implants. We believe that the performance of the framework can be improved by providing additional input information such as fundamental frequency of the harmonic sources to be separated or indeed, midi information related to the various sources.

Acknowledgments: The TITANX used for this research was donated by the NVIDIA Corporation. This work is partially supported by the Spanish Ministry of Economy and Competitiveness under CASAS project (TIN2015-70816-R).

Bibliography

1. Abdel-Hamid, O., rahman Mohamed, A., Jiang, H., Deng, L., Penn, G., and Yu, D. (2014). Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545.
2. Chandna, P. (2016). Audio Source Separation Using Deep Neural Networks, Master Thesis, Universitat Pompeu Fabra.
3. Dong, C., Loy, C. C., He, K., and Tang, X. (2015). Image super-resolution using deep convolutional networks. *CoRR*, abs/1501.00092.
4. Durrieu, J., Ozerov, A., and Févotte, C. (2009). Main instrument separation from stereophonic audio signals using a source/filter model. *17th European Signal Processing Conference.*
5. Gómez, E., Cañadas, F., Salamon, J., Bonada, J., Vera, P., and Cabañas, P. (2012). Predominant Fundamental Frequency Estimation vs Singing Voice Separation for the Automatic Transcription of Accompanied Flamenco Singing. *13th International Society for Music Information Retrieval Conference (ISMIR 2012).*
6. Han, Y. and Lee, K. (2016). Acoustic scene classification using convolutional neural network and multiple-width frequency-delta data augmentation.
7. Hidalgo, J. (2012). Low Latency Audio Source Separation for Speech Enhancement in Cochlear Implants, Master Thesis, Universitat Pompeu Fabra.
8. 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.
9. Kokkinakis, K. and Loizou, P. C. (2008). Using blind source separation techniques to improve speech recognition in bilateral cochlear implant patients. *The Journal of the Acoustical Society of America*, 123(4):2379–90.
10. Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.
11. Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366.
12. Nugraha, A. A., Liutkus, A., and Vincent, E. (2016). Multichannel audio source separation with deep neural networks. Technical report.
13. Rafii, Z., Liutkus, A., and Pardo, B. (2014). *REPET for Background/Foreground Separation in Audio*, pages 395–411. Springer Berlin Heidelberg, Berlin, Heidelberg.
14. Sainath, T. N., Kingsbury, B., and Ramabhadran, B. (2012). Auto-encoder bottleneck features using deep belief networks. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4153–4156. IEEE.
15. Simpson, A. J. R. (2015). Probabilistic Binary-Mask Cocktail-Party Source Separation in a Convolutional Deep Neural Network.

16. Uhlich, S., Giron, F., and Mitsufuji, Y. (2015). Deep neural network based instrument extraction from music. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2135–2139. IEEE.
17. Vincent, E., Gribonval, R., and Fevotte, C. (2006). Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech and Language Processing*, 14(4):1462–1469.
18. Wang, Y., Narayanan, A., and Wang, D. (2014). On Training Targets for Supervised Speech Separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1849–1858.
19. Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701.*