Music Track Separation by using Maching Learning

Carmen Casas Herce
University of Barcelona
Gran Via de les Corts Catalanes, 585
08007, Barcelona
ccasashe86@alumnes.ub.edu

Manuel Andrés Hernandez Alonso
University of Barcelona
Gran Via de les Corts Catalanes, 585
08007, Barcelona
manuelheralo@gmail.com

Abstract

Music is composed of several sources such as voice and instruments, all these sources come together into a mixture and a single wave that we are able to discern separately. Then, the problem at hand is separating the sources into their own independent waveform. Previous research has found different methods based on machine learning and time-frequency domain representation to separate the different sources from the mixture. The main difference between previous work relies on whether to use direct waveform representation for the input of the models or using the time-frequency domain. In this article, we explore the architecture of Open-Unmix, which is based on recurrent layers as the core of the network, that outputs a masking of the original mixture to predict a source stem of the given target instrument. This architecture works in the time-frequency domain to make its predictions. Lastly, the result of implementing this network should result on similar Source-to-Distortion ratios presented on previous research such as F. R. Stöter et. al. *Open-Unmix - A reference implementation for Music Source Separation*[7].

1 Motivation

The music we listen to in our days is a compound of different sounds from a whole diversity of instruments, voices and sound effects. However, we transmit and perceive it as a single wave, so when it comes to dissect its various components from this unique input, it becomes a complex challenge.

This problem has been studied for many years with a lot of different approaches. One of the most common, and one of the first developed was vocal isolation from music, as it involves extracting only a single track and it has multiple applications, as speech denoising or remastering among others. [2]

It is assumed that the final track we use as the input is a compound of N sources. Mathematically, it is represented in the following way [5]:

$$y(t) = \sum_{i=1}^{N} x_i(t) \tag{1}$$

However, separating the different signals is not trivial. They are combined in a non linear way using various processing techniques as reverb and filtering, and the instrument signals are usually highly correlated. [5]

The applications of track separation are huge. Some examples can be: automatic karaoke, DJ software, noise reduction, music transcription or solving the "Cocktail Party Problem" in speech processing, where multiple speakers talk simultaneously.

The way the isolation of tracks is usually done is by using **stems** [3] these are intermediate recordings that mix in a single (stereo) recording all the instruments that are alike. They are divided in: drums

(percussion), bass (involving all the bassline melody), vocals and others (where we have guitar or piano tracks).

Therefore, the aim of the article is to explore different algorithms used in the present to separate a single music track into its different components, as it is represented in the figure 1.

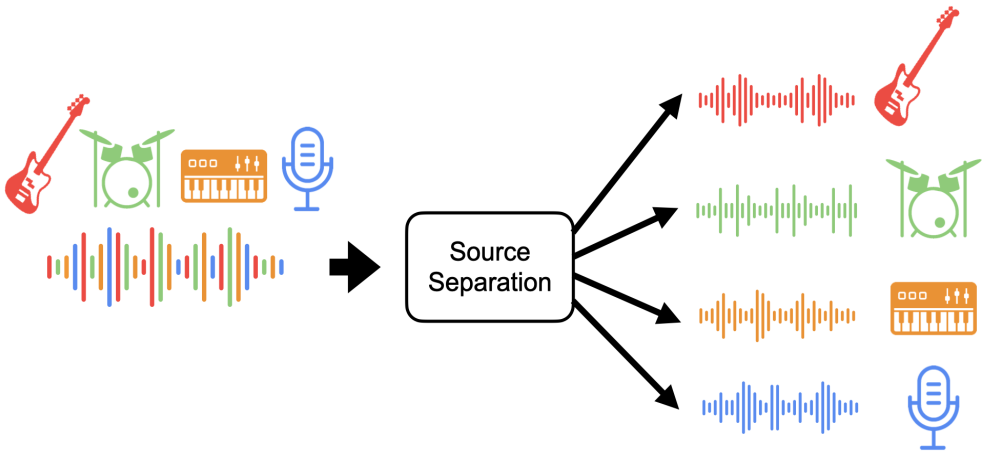

Figure 1: Scheme of the inputs and outputs of a source separation algorithm. [5]

2 State of the art

As previously stated, several approaches have been developed in order to solve this problem. Some authors have tried to separate the different sources directly in the waveform in which the audio is stored. Working directly with the primary waveform, avoiding different transformations, allows the model to exploit all the variables implicitly contained in the raw signal, as the phase. Even though these algorithms were considered less accurate in order to process music signals, lately it has been proved its valid performance comparing to models that are based in other approaches. [4]

For the specific task of music source separation, one of the most well-known algorithms is *Wave-U-Net*, a Convolutional Neural Network adapted from the U-Net algorithm, developed originally for image segmentation. The Wave-U-Net algorithm is able to get outputs from N different sources (instruments and voices) from a single o multi-channel input. [8]

On the other hand, some models preprocess the input using different transformations trying to get a representation in which the multiple sources can be easily separated. It is important to take into account that, when we operate in another representation, we need to be able to recover the original domain in the waveform to get an understandable output.

The most typical transformation is the **Short-time Fourier Transform** (STFT) [5]. This method consists on applying a Fourier Transform to a window of the wave and move it along all the signal. Mathematically, the Fourier Transform is expressed as:

$$f(\omega) = \int_{-\infty}^{\infty} f(x) \exp^{-i2\pi\omega x} \, dx \tag{2}$$

The output function is in the frequency space, so the result of the STFT is a matrix whose axis are the time and the frequency, and whose values are the FT of that window in the different frequencies. The difference between both representations can be seen in the figure 2. There are some free variables in the transformation, as the window and the hop length, which determine both axis commented, and the overlap, a fundamental factor to be able to perform the inverse transformation correctly.

One algorithm based on this type of transformation is the **DeepConvSep** [6]. From the input spectrogram it estimates the spectrograms from each source separately, using a convolutional encoder-decoder architecture. Related work, as can be seen in [3], is also based in convolutional neural networks.

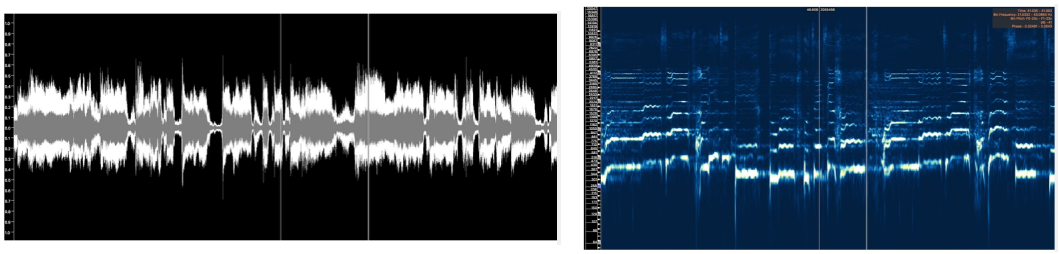

Figure 2: Visualization of the input (left) and output (right) of a signal after a STFT transformation [2]

Summing up, both approaches have been broadly studied and developed in order to perform music track separation in different sources, and a broad range of algorithms have been proven to perform correctly for this task in the recent years.

3 Methods

Firstly, let us recover our problem at hand: separating the combination of the sources from the mixture audio (represented by the equation (1)). For this we have to divide the problem into more digestible sub-problems:

1. What data do we use? And how do we represent it?
2. What architectures are suitable to separate the data in this representation?
3. How do we train this architectures?
4. How do we measure the performance of the models?

Thus, this section is going be dedicated to answering this questions and how to achieve interesting results regarding music source separation.

3.1 Data for music source separation

The data needed for this problem is separated into source components (stems), and mixture data, which is the addition of several stems. This data will be in the waveform, that represents the wave of the different instruments or sources. The main problem this kind of data is that it may have a heavy computational load, as its size is directly correlated with the high sampling rates used for music.

One way to solve this problem is down-sampling the data, which can be used to remove high frequencies when trying out models. Another way would be finding other representations to train the models with. It is important that the new representations keep its ability to represent audio in a separable manner. For the scope of this article, we will focus on the previously mentioned STFT representation of the data, as it keeps the necessary information for separating the sources on the new representation. Additionally, the STFT is invertible, so retrieving the original sources waveform is a given.

Specifically, we can train a model to retrieve the original sources from the STFT via masking, which is a matrix with the same size as the spectrogram that contains values in the interval $[0.0, 1.0]$ inclusive. Each value in the mask determines the proportion of energy that the original mixture that a source contributes. So a value of 1.0 will allow all sound through, and similarly, a value of 0.0 will allow none of the sound through. Finally, we apply the mask through element-wise multiplication of the mask with the spectrogram:

$$S_i = \hat{M}_i \odot |Y| \tag{3}$$

Where S_i represents the i-th source in a mixture represented by a spectrogram $|Y|$, and $\hat{M}_i$ represents the mask for the i-th source.

Another problem that arises when working with the music source separation problem is copyright, making hard to obtain datasets that include multi-track recordings with their respective stems. Luckily, some free to use datasets have been made available by researches and other enthusiasts.

3.2 Machine learning architectures

The main architecture that this article is going to be focusing in is Open-Unmix [7] neural network architecture. This model takes as input the previously mentioned spectrograms and functions in mask based layers where the network progressively separates distinct features using the learned mask filters.

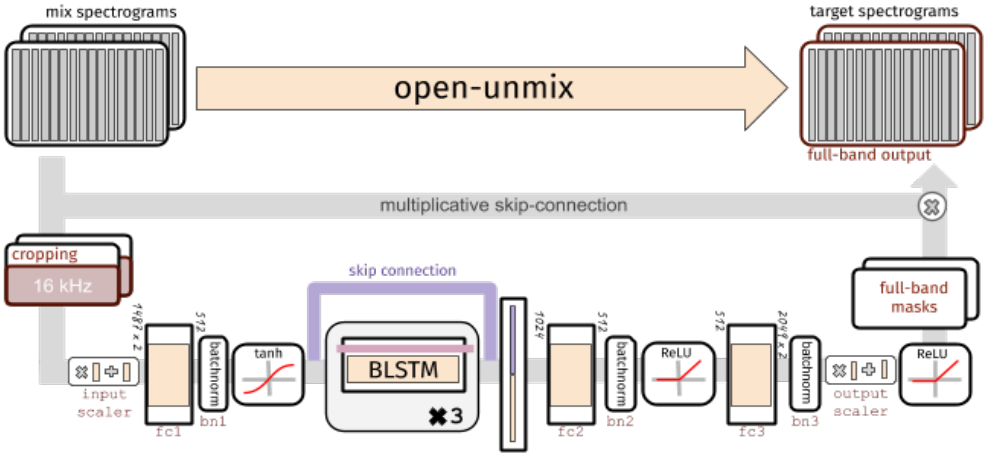

Figure 3: Diagram of Open-Unmix's network architecture. Original image from Fabian-Robert Stöter [7]

Open-Unmix is a convolutional neural network with some recurrent layers. It is based as a variant of the U-Net. Open-Unmix has one fully connected layer with batch normalization and a $tanh$ activation function, then we find three **Bidirectional Long Short-term Memory** (BLSTM) layers, and finally two more fully connected layers with batch normalization and ReLU activations.

To be more precise, the model is separated into several phases:

1. **Input Stage:** As it was mentioned before, the model operates in the time-frequency domain (spectrograms) to perform the predictions. The input is then standardized using the global mean and standard deviation for every frequency bin. Additionally, the batch normalization is used.
2. **Dimensionality reduction:** The BLSTMs are not operating in the original resolution, but after the normalization, the network learns to compress the frequencies and channel axis of the model. This helps the network converge faster.
3. **Bidirectional-LSTM:** This is the core of the network, the BLSTM is used to evaluate data of any length due to its recurrent nature. BLSTM uses information from the past and the future to make predictions on the source spectrogram. Consequently, Open-Unmix with BLSTM can't be used for online in real time predictions, so it is of importance to change the model to an Open-Unmix one with unidirectional LSTMs.
4. **Output stage:** Likewise with the Input stage, the signal is decoded back into the original dimensionality. In the last step, the original input spectrogram is multiplied with the output, so at the end the network learns a mask rather than the direct spectrogram.

One thing to note is that Open-Unmix model is for separating one source from the mixture, for several sources one must train several Open-Unmix models for the different targets. Then, to mix the output spectrograms of the models together one can use a multichannel generalized Wiener filter, before applying the inverse STFT.

3.3 Training the models

Now, to actually use the model for prediction and separation, we need to train the Open-Unmix model. To achieve this we have to define a loss function and the optimizer. For this article, the chosen loss was L1 loss between the original stem and the predicted source given by:

$$L1Loss = \sum_{i=1}^{n} |S_{stem} - S_{predicted}| \tag{4}$$

And for the optimizer of the loss function, the chosen algorithm was the Adam optimizer. This optimizer can me tuned for the given task, but we stuck to the given learning rate of 1.0×10^3. Then, by merging everything together into a training loop where at each step, a given batch of the data is computed and then optimized for in the model. After several epochs of running through the available data, we end up with a trained model ready for separating a source from the rest of the mixture.

It is important to note again that we need to repeat this process for every type of target source we want to separate such as voice, drums, guitar, piano, etc.. Then, it would be necessary to integrate them into a single parallel pipeline as it was mentioned before.

3.4 Measuring performance of the models

Everything until now has been on the theoretical assumption that it will perform as expected, but we haven't really mentioned how to measure how good a source is. Different sources can be similar to their original stems but have artifacts or noise that make them less desirable for any use. Then, there's two types of measures to use for this purpose, objective metrics and subjective metrics. The latter one would be survey based, or by listening the results of the model after it separates the sources from the mixture.

There are several objective measures to check the quality of the audio given by the predicted source: Source-to-Distortion Ratio (SDR), Source-to-Interference Ratio (SIR), and Source-to-Artifact Ratio (SAR). All of these evaluations assume that a predicted source $\hat{s}_i$ is actually composed of four elements relating to interference e_{int}, noise e_{noise}, and artifacts e_{art}, as well as the original target source s_i:

$$\hat{s}_i = s_i + e_{int} + e_{noise} + e_{art} \tag{5}$$

The actual calculation of the elements is complex and not an important focus for this article, for more information on the topic we refer to the original paper from Emmanuel Vicent et. al. [1].For the scope of this article, we'll see the SDR evaluation that is given by:

$$\mathbf{SDR} := 10 \log_{10} \left(\frac{||s_i||^2}{||e_{int} + e_{noise} + e_{art}||^2} \right) \tag{6}$$

One last thing to note is that SDR is just an approximation of how good a given predicted source will sound, but it may be that two separate models with the same SDR values sound differently. Consequently, it is important to revise the results from the model by reproducing the audio from the predicted sources as it was mentioned before.

4 Conclusions

In this article we have explored the topic of Music Track Separation using Machine Learning. This broadly-studied subject has several approaches that we have analyzed and compared, in order to perform a deep understanding of the most interesting algorithm found in the literature.

First of all, it is essential to consider how the source data is conveyed as input to the model. One approach is to use the raw signal in its waveform, allowing for a direct input. This methodology has different advantages, as there is no loss of information during the preprocess and obtaining an output that is directly understandable.

However, transforming the original signal by using a Short-time Fourier Transform is widely extended. Although it requires to preprocess the input and the output data, it is very convenient to use because the frequency space is able to capture the pitch, timbre, and harmony of the different sources that compound the wave, essential properties to perform correctly the Music Track Separation.

Finally, we have focused on the Open-Unmix algorithm, whose architecture works by means of recurrent layers. This model ends up learning a mask to retrieve the target source out of the original waveform of the mixture. The final predicted sources are then evaluated with Source-to-Distortion ratios as an objective measure, but also opening the file to listen to the final results as a subjective comparison, as the output should be easily understandable. Although it hasn't been implemented in this study due to a lack of computational power, by implementing and training the described architecture of the pipeline one should end up with similar results to previous research.

References

[1] R. G. Emmanuel Vicent and C. Fevotte. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing*, 2006.

[2] A. Koretzky. Audio ai: isolating vocals from stereo music using convolutional neural networks. *Towards Data Science*, 2019. `https://acortar.link/5TbnA1`.

[3] A. Koretzky. Audio ai: Isolating instruments from stereo music using convolutional neural networks. *Towards Data Science*, 2019. `https://acortar.link/QFD4OY`.

[4] F. Lluís, J. Pons, and X. Serra. End-to-end music source separation: is it possible in the waveform domain?, 2019.

[5] E. Manilow, P. Seetharman, and J. Salamon. *Open Source Tools & Data for Music Source Separation*. https://source-separation.github.io/tutorial, Oct. 2020. URL `https://source-separation.github.io/tutorial`.

[6] M. Miron, P. Chandna, G. Erruz, and H. Martel. *DeepConvSep*. Music Technology Group, Universitat Pompeu Fabra, 2017. URL `https://github.com/MTG/DeepConvSep`.

[7] 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. doi: 10.21105/joss.01667. URL `https://doi.org/10.21105/joss.01667`.

[8] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.