Music Source Separation

A Music Information Retrieval Project

https://github.com/alessandromanattini/MusicSourceSeparation

Farnoosh Afshinrad Giorgio Magalini Alessandro Manattini
Mattia Montanari

Politecnico di Milano
Academic Year 2024/2025

Abstract

This project explores one of the most significant and challenging areas within the discipline of **Music Information Retrieval (MIR)**: **source separation**. Source separation refers to the process of isolating individual sound sources from a mixture, such as extracting vocals, drums, or harmonic instruments from a complete musical track. This task plays a crucial role in various applications, including **music production**, **remixing**, **audio restoration**, and **music analysis**.

We begin by introducing the fundamental principles of source separation, aiming to clarify its goals and the complexity of the problem. Following this, we examine and compare a range of techniques developed over time, starting with **traditional signal processing approaches**—such as Non-negative Matrix Factorization (NMF)—and moving toward contemporary methods based on **machine learning** and **deep neural networks**.

Throughout the report, we will apply appropriate **evaluation metrics** to assess the performance of each method, such as the Signal-to-Distortion Ratio (SDR) and the Signal-to-Interference Ratio (SIR). These metrics allow us to quantify the progress made in this field and highlight the **strengths and limitations** of each approach. By doing so, we aim to provide a clear picture of how source separation has evolved and where it is heading in future research and real-world applications.

Contents

Introduction

Source separation refers to the set of techniques used to isolate individual sound sources from a mixture—typically a musical recording where multiple instruments and voices are playing simultaneously. The goal is to extract one or more of these sources (such as a vocal track, drums, or a solo instrument) while minimizing interference from the others.

This task is inherently complex due to several interrelated factors. First, the acoustic characteristics of each instrument—such as timbre, pitch, note duration, and harmonic content—can vary widely. Second, musical performance introduces variability in timing, dynamics, and articulation. Third, the recording process itself—including microphone placement, room acoustics, and mixing practices—further alters the raw sound of the sources. Finally, post-processing steps such as equalization, compression, reverb, and other effects applied during mixing and mastering can blend the sources in ways that obscure their individual identities.

Because of this complexity, different approaches to source separation have been developed depending on the specific needs of the application. Some tasks are relatively simple, while others are extremely challenging.

At the most basic level, one might separate the stable, harmonic components (such as sustained notes from strings or keyboards) from transient, percussive elements (like drum hits or note attacks). This process is known as harmonic/percussive separation, and it leverages the distinct time-frequency characteristics of harmonic and transient content.

A more difficult task is solo/accompaniment separation, which involves identifying and isolating the main melodic instrument (e.g., a lead guitar or saxophone) from the background accompaniment. This is harder because the solo instrument often overlaps in frequency and time with other instruments, and it may not always be clearly distinguishable in the mix.

An even more complex scenario is vocal separation, where the aim is to isolate the singing voice—male or female—from the rest of the musical ensemble. This task is especially challenging because voices vary greatly in style, range, and timbre, and are often heavily processed with effects like reverb or auto-tune that blend them into the mix.

The most ambitious goal in source separation, and the focus of this project, is

full stem separation—that is, extracting every individual source from a mixture to produce isolated tracks (or "stems") for each instrument and vocal part. This is considered the hardest form of source separation, as it requires resolving overlapping spectral content and maintaining audio quality for each component.

Research in source separation is driven by a wide range of real-world applications:

- Audio Remixing: By isolating individual tracks, sound engineers or DJs can adjust levels, change effects, or creatively rework existing music without access to the original multitrack recordings.
- Audio Upmixing: This involves transforming a mono or stereo recording into a multichannel format (e.g., 5.1 surround sound) for enhanced spatial presentation, such as adapting music or dialogue for cinema playback. Source separation helps distribute sounds across channels in a way that enhances listener immersion.
- Music Analysis and Transcription: Tasks like beat tracking, chord recognition, and automatic transcription benefit greatly from source separation, since analyzing a clean, isolated source (e.g., just the drums or just the piano) reduces ambiguity and increases accuracy.
- Music Education: Source separation can aid musicians who are learning a specific part in a piece. For example, a student learning the bass part can isolate it from a recording, slow it down, or mute it to play along as if performing with the full ensemble.

These examples illustrate the broad relevance of source separation, not only as a technical challenge but also as a powerful tool for creative, analytical, and educational purposes.

Dataset

For this project, we selected the MUSDB18-HQ dataset. It is the uncompressed, high-quality version of the widely used MUSDB18 dataset, which is distributed in a lighter format using MP4 (AAC compression). MUSDB18-HQ provides higher audio fidelity by storing all files in uncompressed WAV format, making it more suitable for tasks where audio quality is crucial, such as source separation. The dataset contains 150 full-length stereo tracks spanning a variety of musical genres. Each track includes the mixture as well as four isolated stems:

- drums.wav
- bass.wav
- other.wav
- vocals.wav

All files are stereophonic and encoded at 44.1 kHz with 24-bit depth. The dataset is divided into two subsets:

- A training set (train/) with 100 songs
- A test set (test/) with 50 songs

MUSDB18-HQ was developed to serve as a reference database for the development and evaluation of source separation algorithms.

Traditional Approaches

As far as concern the traditional signal processing approaches we adopted two different approaches: one based on harmonic percussive separation adopting different techniques for obtain all the stems with in particular sim-repet for vocal extraction, and a different approach based on NMF to obtain directly all the stems in one time

3.1 Custom approach

After loading the dataset, we first did an harmonic percussive separation to differentiate the harmonic part from the percussive elements in the mixture. In this way we obtain the components *drums*. Then from the harmonic part we extract the vocal using the REPET-SIM technique obtaining *vocals*. To compute *bass* we just apply to the harmonic mixture just computed a Butterworth filter and finally *other* is simply computed by subtraction starting from the original mixture and subtracting *drums*, *vocals* and *bass*. In particular the steps of the algorithm followed are:

1. **HPSS (Harmonic–Percussive Source Separation):** Decompose the input STFT into harmonic and percussive spectrograms using tunable margin and power parameters.
2. **Drums Cleaning:** Split the percussive spectrogram into magnitude and phase, attenuate the 300–3000 Hz band by a user-defined factor to remove vocal leakage, and reconstruct the cleaned drum waveform via inverse STFT.
3. **Bass Extraction:** Apply a low-pass Butterworth filter (configurable cutoff frequency and filter order) to the harmonic waveform to isolate bass frequencies, ensuring the cutoff remains below the Nyquist limit.
4. **Vocals Extraction (REPET-SIM):** Compute a cosine-similarity matrix across harmonic spectrogram frames, select similar frames for each time index, apply median filtering to estimate the repeating background, subtract it to highlight non-repetitive vocal content, and reconstruct via inverse STFT.
5. **Residual ("Other") Calculation:** Subtract the reconstructed drums, bass, and vocals from the original time-domain signal; the remainder constitutes the "other" channel.

Here follow a small digression regarding the HPSS and REPET-SIM algorithm and how they work.

3.1.1 REpeating Pattern Extraction Technique with SIMilarity (REPET-SIM)

REPET-SIM is a source separation algorithm designed to separate repeating background elements (typically music accompaniment) from non-repeating foreground elements (like vocals or solos) in a mixture. It builds upon the original REPET technique, which assumes that the background music is periodic—repeating over time.

However, REPET-SIM goes beyond strict periodicity by using similarity measures to find quasi-repetitions rather than requiring exact ones. This makes it more flexible and effective on real-world music where the accompaniment isn't perfectly periodic.

In few words the algorithms is:

- STFT: first compute the spectrogram V of the mixture.
- Similarity Matrix: Calculate a cosine similarity matrix between all time frames of the spectrogram.
- Repeating Frame Selection: For each time frame, identify a set of most similar frames thanks to the similarity matrix. These represent other points in time where similar background content occurred.
- Median Filtering: For each time-frequency bin, replace its value with the median across the selected similar frames. This filters out non-repetitive foreground elements (like a voice) while preserving the consistent, repeating background. To say it in other words, you isolate a time frame t_1, thanks to similarity matrix you find all the other parts in the song really similar to it because the accompaniment is the same (just the melody on top of the background is different), and then substitute the content of this time frame with the median across all the similar time frames filtering out the only different parts (i.e. the foremelody). In this way we obtain an estimation for the spectrogram of the background.
- Foreground Estimation: Subtract the background spectrogram just computed from the starting spectrogram of the mixture.
- iSTFT: To reconstruct the time-domain signals for both background and foreground.

3.1.2 Harmonic Percussive Source Separation (HPSS)

Harmonic Percussive Source Separation (HPSS) is a digital signal processing (DSP) technique that decomposes an audio signal into its two fundamental components: **harmonic** and **percussive** sounds.

The method works by analyzing the signal's **spectrogram**, a visual representation of frequency over time. In this view, different sounds have distinct shapes:

- **Harmonic content**, like melodies or vocals, has a stable pitch and forms **horizontal structures**.
- **Percussive sounds**, like drum hits or claps, are short, transient events that span many frequencies at once, creating **vertical structures**.

A common HPSS algorithm exploits this by applying **median filters** to the spectrogram: one horizontally to isolate harmonic structures and one vertically to isolate the percussive transients. From these filtered views, time-frequency **masks** are generated to separate the original audio signal.

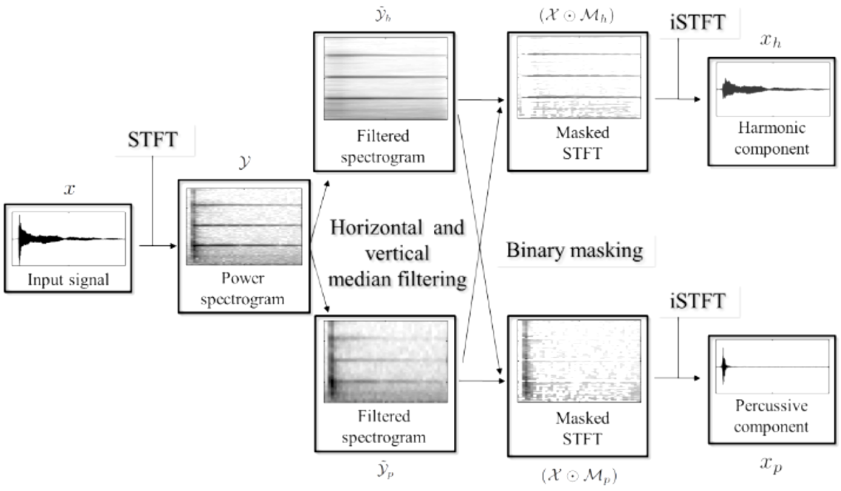

Figure 3.1. HPSS scheme

3.2 NNMF approach

The second way consist in doing once again a harmonic percussive source separation and applying on harmonic part just extracted a standard NMF approach.

The algorithm performs source separation by applying non-negative matrix factorization (NNMF) to frequency-masked bands of the mixture's magnitude spectrogram. After converting the input to mono and computing its STFT, the algorithm constructs frequency masks for drums, bass, vocals and "other" based on predefined ranges. An initial HPSS step guides separation by splitting percussive and harmonic content. NNMF is then run on each masked band—using a configurable number of components—to estimate the individual source spectrograms. Soft-masking mitigates artifacts, and inverse STFT plus normalization reconstruct each time-domain stem.

1. **Preprocessing and STFT:** Convert input to mono, compute STFT with specified `n_fft` and `hop_length`, and extract magnitude (S_{full}) and phase.

2. **Frequency Mask Creation:** Define boolean masks for drums, vocals, bass and other based on their frequency ranges.

3. **Initial HPSS:** Decompose S_{full} into percussive and harmonic components ($S_{\text{percussive}}$, S_{harmonic}) to guide subsequent separation.

4. **Band-wise NNMF Separation:**

   - Apply NNMF to $S_{\text{percussive}}$ masked for drums.
   - Apply NNMF to S_{harmonic} masked for bass.
   - Subtract bass estimate and apply NNMF to the residual harmonic for vocals.
   - Subtract bass and vocals and apply NNMF to the remaining harmonic for "other."

5. **Soft-Masking:** Compute power-based soft masks for each estimated component to reduce bleed and artifacts, then multiply by S_{full}.

6. **Reconstruction:** Combine each masked magnitude with the original phase, perform inverse STFT, and normalize to prevent clipping.

Here a digression on how the NNMF algorythms works

3.2.1 Non-negative Matrix Factorization (NNMF)

The main idea behind Non-negative Matrix Factorization (NNMF) is to decompose a non-negative matrix V, such as the magnitude spectrogram of an audio mixture, into the product of two non-negative matrices: V = WH Where V is the input spectrogram, W is the basis matrix and H is the activation matrix. Each column of W represents a spectral basis or template—essentially capturing the timbre or spectral signature of an individual sound source or component (e.g., an instrument). Each row of H indicates how active that basis is over time—capturing the temporal evolution of each source. The non-negativity constraint is crucial: it aligns with the physical nature of audio magnitude spectrograms which can't have negative energy. The algorithm basically require three steps:

- Initialization: The matrices W and H are randomly initialized with non-negative values.
- Iterative Update: Using multiplicative update rules, the algorithm alternates between updating W and H to minimize a cost function that measures the divergence between the original matrix V and its approximation WH. Common cost functions include Euclidean distance and Kullback-Leibler (KL) divergence
- Convergence: The process continues until convergence is achieved (e.g., the change in cost function is below a threshold).

Once the factorization is complete, you can isolate individual sources from the mixture by selecting the corresponding components (columns in W, rows in H) associated with that source. Multiply these selected basis and activation parts to form a partial spectrogram. Finally, to retrieve the time-domain signal of this source, just apply the inverse short-time Fourier transform (iSTFT) to convert the source spectrogram back to audio.

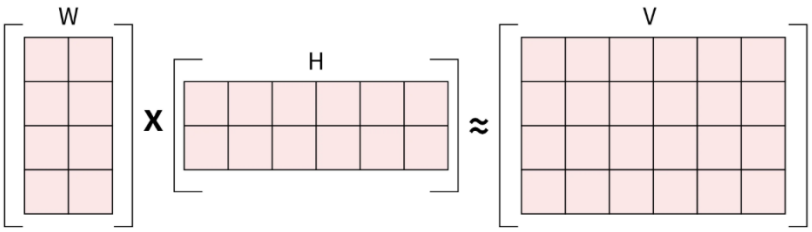

Figure 3.2. NNMF scheme

Novel Approaches

4.1 U-Net

The U-Net is a convolutional neural network (CNN) originally developed for medical image segmentation, but it has also found application in the context of source separation, where it operates not on traditional images, but on audio spectrograms. U-Net is an encoder-decoder network with skip connections, and its structure resembles the letter "U" (hence the name U-Net). It is a supervised model, which learns during training from labeled data. In particular, training involves a set of mixture spectrograms used to feed the network (e.g., vocals + instruments), and for each of them, a target is provided — namely, the clean spectrogram of a single source (e.g., only vocals). During training, the model learns to minimize the difference between its output (the predicted spectrogram of the extracted source) and the ground truth spectrogram, using loss functions such as L1/L2 loss on the magnitude spectrograms, SDR-based loss, etc. To simply describe the architecture:

The encoder, or the left side of the U, compresses the input by applying convolutions followed by max pooling. This helps capture more abstract, high-level features of the sound. Essentially, the deeper you go, the more you lose local details (such as the precise peaks in frequency), but you gain a better understanding of the overall structure of the content (e.g., "here there is voice," "here there is a drum").

The decoder, or the right side of the U, reconstructs the signal using transposed convolutions (also called up-convolutions). Starting from the abstract and compressed representation, it reconstructs a spectrogram — which can be either the target spectrogram of the source or a mask to be multiplied with the input mixture spectrogram.

The skip connections are direct links between corresponding layers of the encoder and decoder. They help retain the local details that may have been lost during downsampling. In fact, when reconstructing the spectrogram, the network also needs specific information, such as the exact shape of spectral peaks or fine temporal/frequency variations, which might be lost in the deeper (more abstract) layers. Essentially, while the decoder is trying to reconstruct the content, the skip connections provide a kind of shortcut, supplying the precise local information preserved on the way down.

In particular in our implementation we created and trained 4 models, one per target

(*vocals*, *drums*, *bass*, *others*) in order to be able to predict source separations for each type of source.

Our implementation of the U-Net architecture integrates convolutional neural networks (CNNs) with recurrent LSTM layers to perform audio source separation on spectrogram representations. The design follows a classical encoder-decoder structure with skip connections, enhanced by temporal modeling to capture sequential dependencies in the audio signal.

Encoder (Downsampling Path) The encoder consists of three blocks, each progressively increasing the number of channels. Each block includes two convolutional layers followed by batch normalization and ReLU activation:

- `Conv1D` → `BatchNorm` → `ReLU` → `Conv1D` → `BatchNorm` → `ReLU`
- `MaxPool1D` is applied between blocks for downsampling.
- Channel progression: `n_bins` → 256 → 512 → 1024

Bottleneck with Temporal Modeling At the bottleneck, the number of channels is expanded to 2048 via a convolutional layer. This is followed by two layers of bidirectional LSTM, which effectively model temporal dependencies across time frames. This stage is essential for capturing long-term musical patterns and dynamics.

Decoder (Upsampling Path) The decoder mirrors the encoder with three upsampling blocks. Each block includes:

- `ConvTranspose1D` for upsampling
- Skip connections from the corresponding encoder layers to preserve spatial detail
- Interpolation where necessary to handle dimensional mismatches

The decoder progressively reduces the number of channels to match the original input spectrogram dimensions.

Output and Masking Strategy The final layer generates a multiplicative mask applied element-wise to the input spectrogram. This technique preserves the original structure while isolating the target source.

Component	Purpose	Benefit
Skip Connections	Preserve fine-grained details	Improve reconstruction quality
LSTM Layers	Model temporal dependencies	Capture musical structure
Multiplicative Masking	Source-aware filtering	Maintain audio coherence
BatchNorm + ReLU	Stabilize training	Accelerate convergence

Table 4.1. Key components and their benefits in the architecture.

Key Features

Input/Output

- **Input:** Spectrogram tensor of shape (`batch`, `time`, `frequency`)
- **Output:** Estimated source spectrogram with same dimensions
- **Process:** The model learns to estimate optimal masks that isolate a given instrument from the mixture.

This hybrid CNN-LSTM architecture effectively combines spatial feature extraction with temporal sequence modeling, demonstrating strong performance in complex tasks like music source separation.

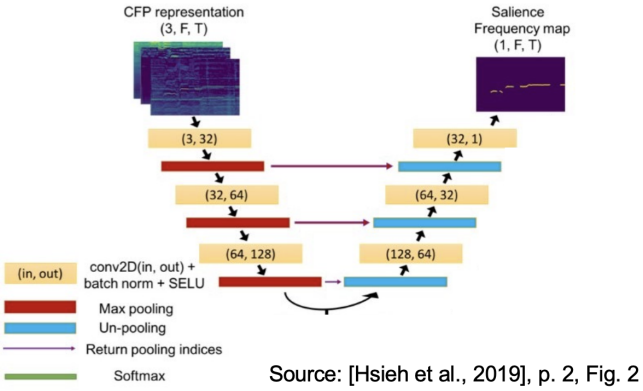

Figure 4.1. U-Net architecture

4.2 Demucs (Deep Extractor for Music Sources)

Speaking about the state of the art, Demucs (Deep Extractor for Music Sources) is one of the most advanced and widely used models for music source separation. Its goal is to isolate vocals, drums, bass, and other instruments from an input mixture. It is a supervised model: during training, it learns from pairs of audio mixtures and their corresponding isolated source waveforms, which serve as ground truth for evaluating the model's predictions.

Demucs is based on a U-Net architecture, but unlike classical U-Nets that operate on spectrograms, Demucs works directly in the time domain. In other words, it takes raw audio waveforms as input instead of converting them into a frequency representation.

Just like a typical U-Net, Demucs uses a U-shaped encoder-decoder structure with skip connections between symmetrical layers of the encoder and decoder. But since the input is now a 1D audio waveform (samples over time), the convolutions are 1D convolutions.

More precisely, a 1D convolution applies multiple filters that slide along the time axis. For example, if a layer uses 64 filters, each filter produces a new output sequence—this is what we refer to as a new channel. However, unlike the original audio channels (e.g., mono or stereo), these new channels don't have a physical meaning. Instead, each channel is the activation map of a learned filter—some may respond to percussive sounds, others to harmonic content, and so on.

A key feature of these convolutions is the use of stride grater than one. The stride determines how much the convolution window shifts at each step. A stride greater than one means some samples are skipped, leading to a reduction in temporal resolution. The signal becomes shorter at each layer, just like what happens in image processing with max pooling.

So, the deeper you go into the encoder, the shorter the signal becomes in time, but the number of channels increases. This lets the network capture increasingly abstract and complex features—trading off precise local details for higher-level understanding (e.g., detecting "a voice" instead of "this exact frequency peak").

The decoder then performs the inverse operation. It uses 1D transposed convolutions (also known as deconvolutions) to upsample the signal, gradually restoring the original temporal resolution. Crucially, the decoder directly outputs the separated waveform for each source—meaning it produces raw audio at the same length as the input. This removes the need for any intermediate step like iSTFT (used to convert spectrograms back to audio).

Finally, skip connections play the same essential role as in classic U-Nets: they pass fine-grained temporal details from the encoder directly to the decoder. This helps preserve important low-level information (like transients, sharp attacks, and subtle timing cues) that could otherwise be lost during downsampling.

Anyway the biggest innovation introduced in the DEMUCS structure respect a standard U-Net is the Bidirectional Long Short-Term Memory (BLSTM) layer. This part of the network is often referred to as the bottleneck because it's located at the narrowest (most compressed) point of the U-shaped architecture, which is the between the encoder and the decoder where the encoder produces a compressed representation of the audio. Here we have a representation shorter in time, but richer in features and which mostly contains local information from the receptive fields of the convolutional layers (i.e., what each filter "sees" at a given point). Music, though, isn't just about local patterns. It's something definitely structured over time: melodies evolve, rhythms repeat, phrases build on one another. To model this, the network needs to understand relationships across long time spans—beyond what convolution alone can capture.

This is where the BLSTM comes in: A Bidirectional LSTM is a type of recurrent neural network that is designed to model sequences by remembering long-range dependencies, which in particular reads the input both forward and backward in time (from here the name 'Bidirectional'). In this way it achieve more meaningful results since it can consider past and future context simultaneously. In Demucs, this helps the network capture musical structure: like a sustained vocal note, a drum fill, or a buildup over several seconds.

To sum up: we can say that the encoder breaks the signal down into abstract pieces, The BLSTM then connects those pieces over time, learning how they relate musically and finally the decoder uses this enriched, temporally-aware representation to reconstruct each individual source more accurately.

In this project, we used a pretrained model from the Hybrid Demucs (HDemucs) family, trained on the MUSDB18 dataset. This model is ready for inference—meaning it can perform source separation directly, without the need for retraining. Our goal was simply to evaluate the results achievable with one of the most advanced neural networks currently available for music source separation.Unlike the classic Demucs architecture, which operates solely in the time domain on raw audio, the hybrid version combines both time-domain and frequency-domain processing. By leveraging the strengths of both representations, HDemucs enhances the quality of the separation.

Metrics

To evaluate how effectively the model isolates individual sources from a mixture, we relied on a family of metrics known as BSS Eval metrics (from Blind Source Separation evaluation). In particular:

- **SIR (Signal-to-Interference Ratio)** focuses on how well the model removes interference from other sources. It indicates how effectively the target source is isolated from the rest of the instruments or voices in the mixture. A high SIR means that unwanted elements—like drums bleeding into a vocal track—have been successfully suppressed.

$$\mathrm{SIR} = 10 \log_{10} \left(\frac{\|s_{\mathrm{target}}\|^2}{\|e_{\mathrm{interf}}\|^2} \right)$$

- **SAR (Signal-to-Artifacts Ratio)** measures the amount of unnatural distortion (or artifacts) introduced by the model during separation. These artifacts can include warbling, metallic sounds, or phase inconsistencies. A high SAR score indicates that the separated signal sounds clean and natural, without noticeable glitches introduced by the processing.

$$\mathrm{SAR} = 10 \log_{10} \left(\frac{\|s_{\mathrm{target}} + e_{\mathrm{interf}}\|^2}{\|e_{\mathrm{artif}}\|^2} \right)$$

- **SDR (Signal-to-Distortion Ratio)** is the most widely used and general-purpose metric for evaluating separation quality. It compares the energy of the clean, reference signal to the energy of all forms of distortion combined—interference, artifacts, and noise. In essence, SDR provides a single score that reflects the overall quality of the separated output. Higher SDR values correspond to better separation with minimal distortion.

$$\mathrm{SDR} = 10 \log_{10} \left(\frac{\|s_{\mathrm{target}}\|^2}{\|e_{\mathrm{interf}} + e_{\mathrm{noise}} + e_{\mathrm{artif}}\|^2} \right)$$

5.0.1 Evaluation Metrics: Using `mir_eval` for Source Separation Quality

To compute these metrics in practice, we used the `mir_eval` library, specifically the `separation.bss_eval_sources` function. This function takes as input the reference sources and the corresponding estimated sources—both represented as NumPy arrays with shape `(n_sources, n_samples)`—and automatically handles the evaluation. One crucial aspect of the implementation is that it internally finds the optimal permutation of the estimated sources to match the reference sources, ensuring that the metrics are not penalized due to differences in source ordering.

For each source, the function decomposes the estimated signal into four components: the correctly estimated target signal, the interference from other sources, the noise, and the artifacts introduced by the model. Based on the energy ratios between these components, the function returns the SIR, SAR, and SDR values for each individual source. The use of `mir_eval` thus simplifies the evaluation process while ensuring that the results are consistent with widely accepted standards in the source separation community.

Results Comparison

6.1 Traditional Approches

Here follow the results in terms of SDR, SAR, and SIR for the source separation achieved using traditional approaches, in particular one technique based on harmonic percussive source separation and repet-sim for vocals.

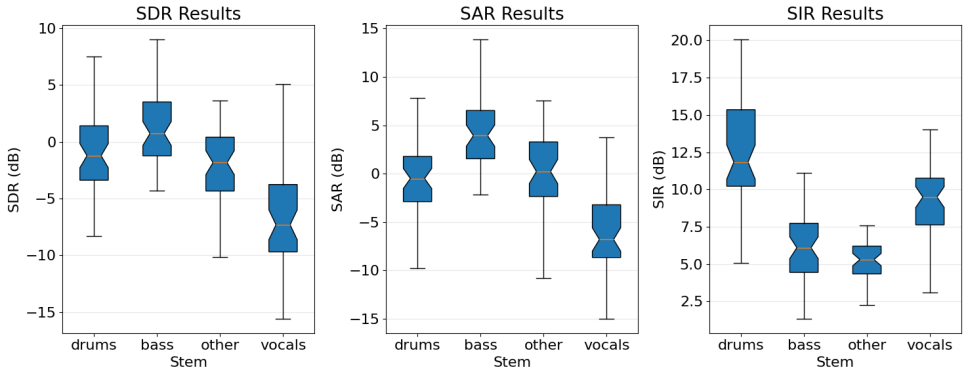

Figure 6.1. HPSS+REPET-SIM

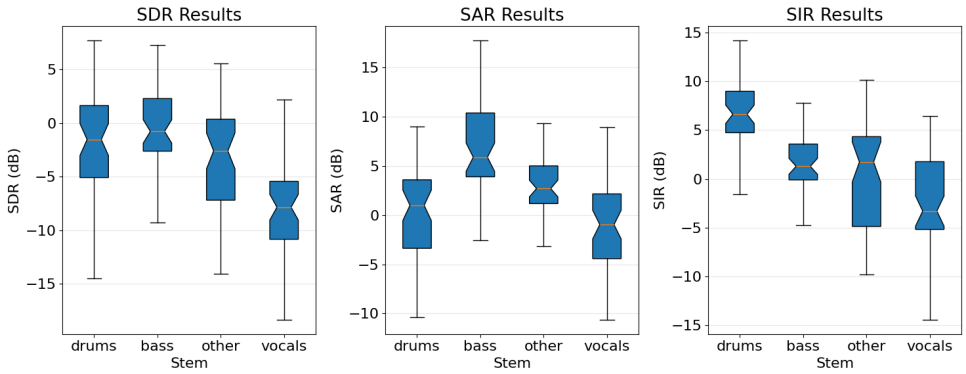

Figure 6.2. HPSS+NNMF

The comparison between the two configurations clearly shows that HPSS + REPET-SIM outperforms HPSS + NNMF on all three parameters (SDR, SAR, SIR) and for all stems:

- SDR (overall quality): higher average values in REPET-SIM, indicating more faithful separations.
- SAR (number of artifacts): REPET-SIM maintains fewer artifacts, while NNMF suffers from high distortion especially on "vocals" and "other".
- SIR (interference suppression): REPET-SIM achieves high SIRs in all stems, while NNMF often experiences negative SIRs, especially on "other" and "vocals".
- **Conclusion**: HPSS + REPET-SIM ensures significantly better separation (less artifacts and higher signal isolation) compared to the NNMF-based approach.s at identifying and separating repeating rhythmic patterns. However, this specialization results in lower overall quality (SDR) and more processing artifacts (SAR), particularly for vocals, which are separated poorly.

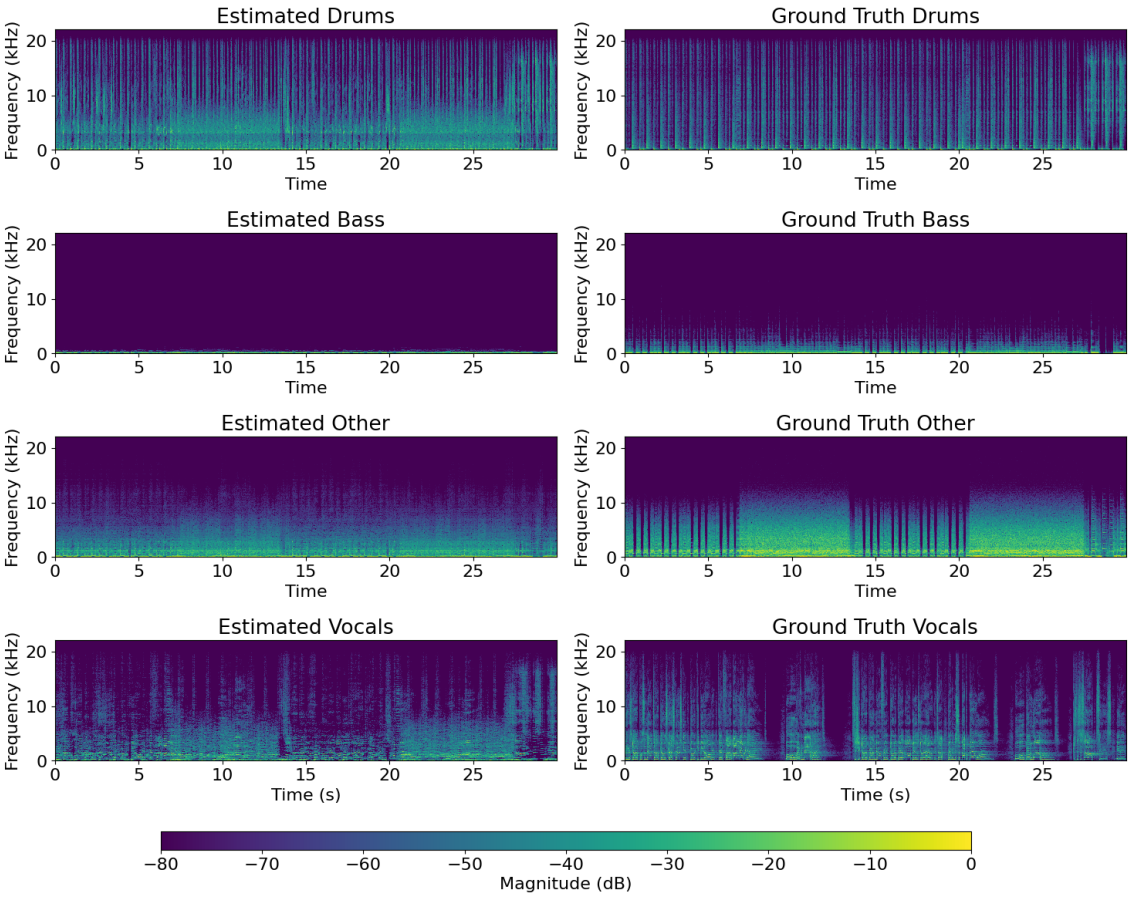

Figure 6.3. Spectrograms Comparison of HPSS + REPET-SIM

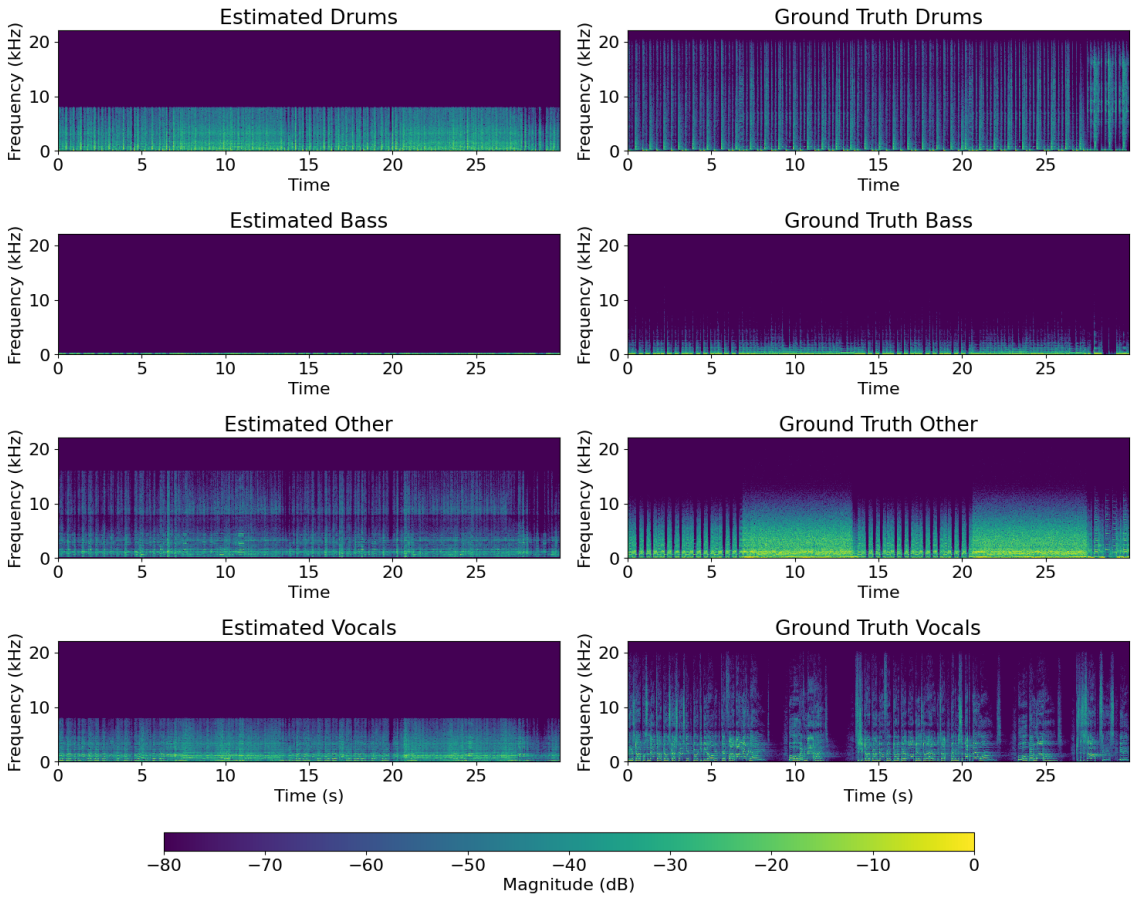

Figure 6.4. Spectrograms Comparison of NNMF

6.2 Novel Approaches

We started this project with the idea of establishing a performance spectrum for source separation: at the lower bound, we aimed to demonstrate the limitations of traditional signal processing techniques, while at the upper bound, we evaluated the capabilities of state-of-the-art models. Our goal was to position the results obtained from a neural network we designed and trained ourselves somewhere in between these two extremes.

As reflected in the results, the performance of our U-Net–like architecture was ultimately unsatisfactory and did not meet our expectations. Several factors may account for this underperformance.

First, the architecture may have been excessively large relative to the dataset. While the dataset itself was reasonably substantial, the model's size likely led to overfitting or inefficient training. Moreover, our implementation followed a relatively straightforward approach without dedicated optimization. Time constraints may have contributed to suboptimal choices regarding hyperparameters, loss functions, and convergence strategies.

Another important limitation was the absence of data augmentation, which likely reduced the model's ability to generalize to unseen data.

Despite these challenges, developing a custom model provided valuable insights into the source separation problem. The clear performance gap between our network and advanced pretrained systems such as HDemucs underscores the advantages offered by large-scale training, architectural sophistication, and the use of extensive curated datasets in achieving state-of-the-art results.

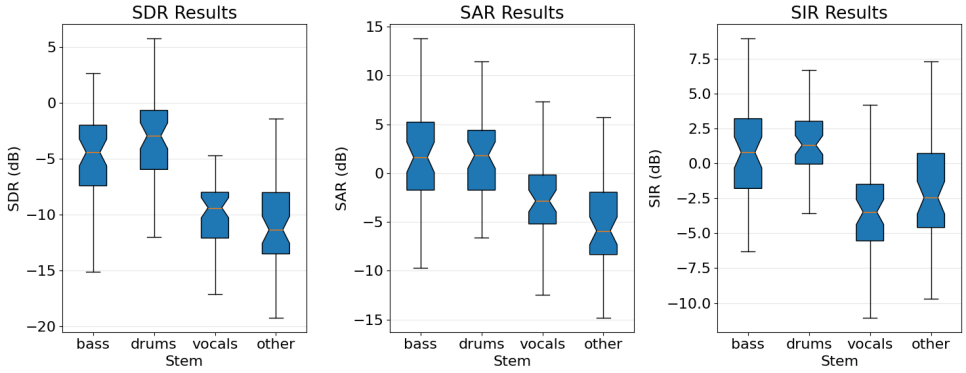

Figure 6.5. U-Net

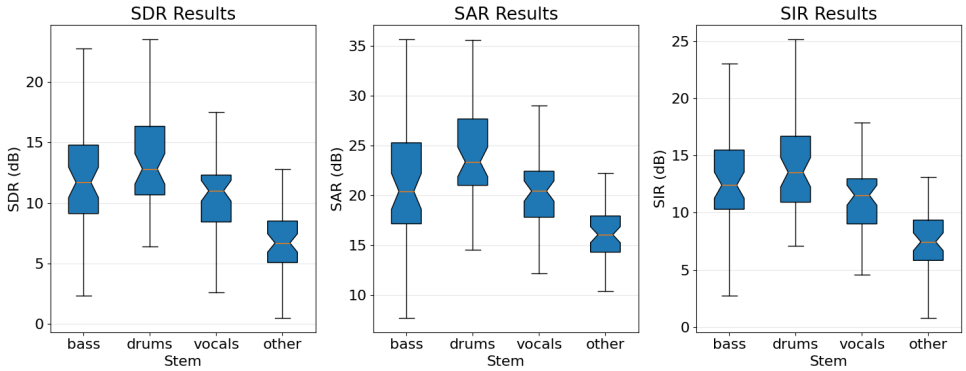

Figure 6.6. Demucs

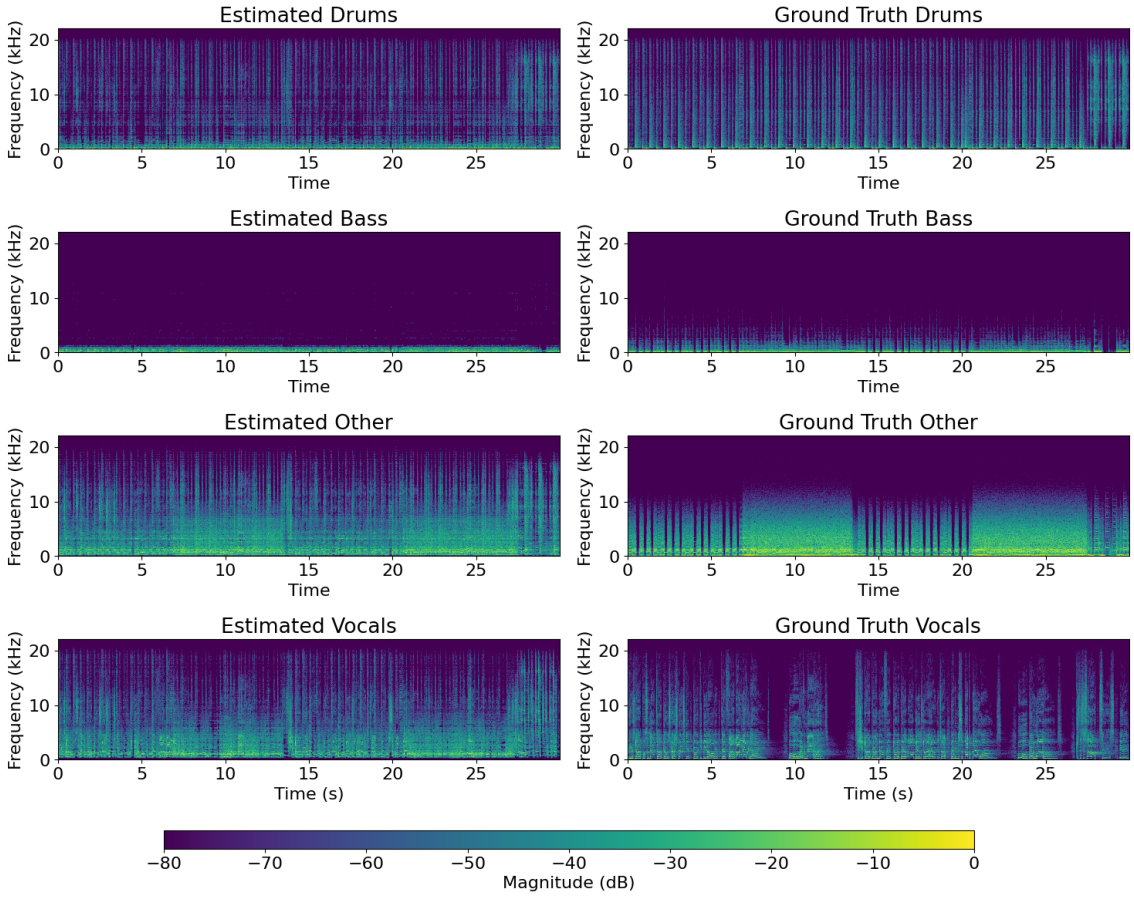

Figure 6.7. Spectrograms Comparison of U-Net

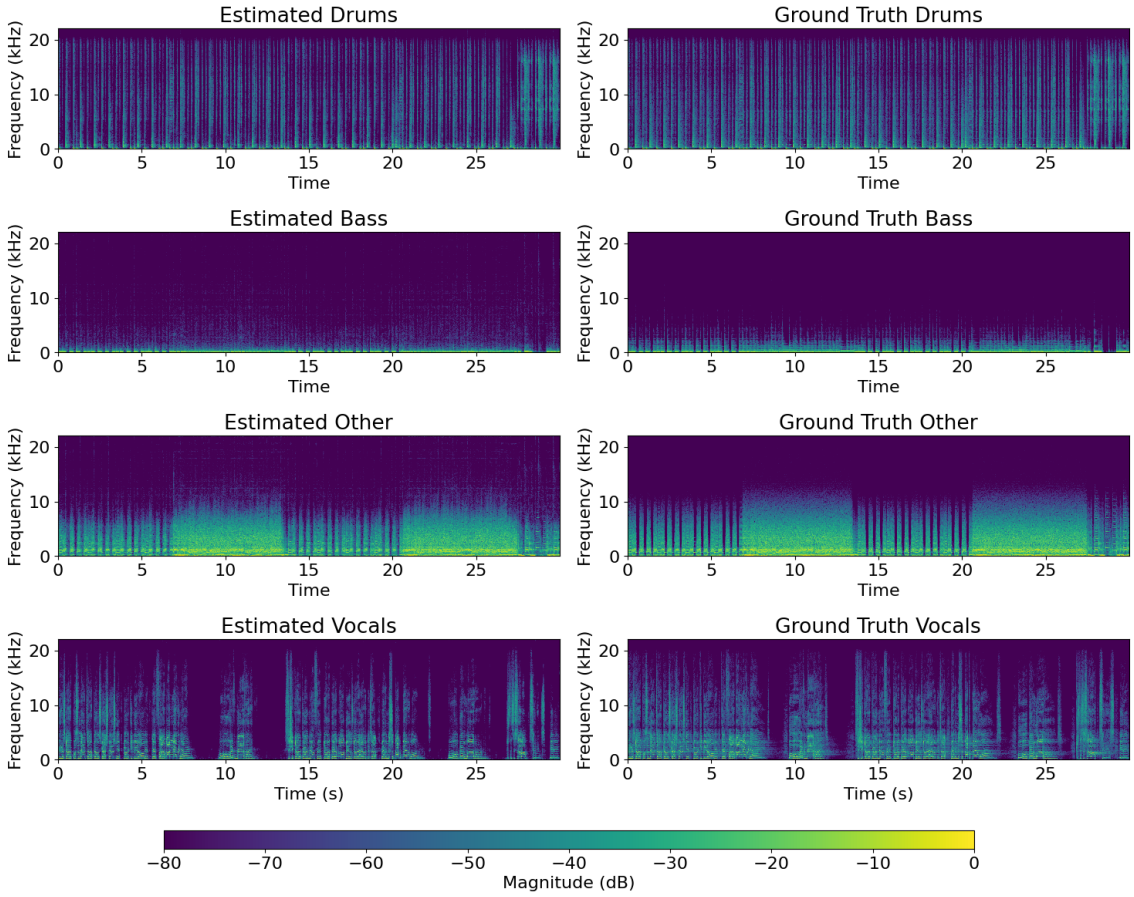

Figure 6.8. Spectrograms Comparison of Demucs

Extra: Simplified Demucs

Since we were not able to achieve acceptable results with our own neural network implementation, we explored different strategies during the design phase. The most intuitive starting point was a simplified U-Net–like architecture, but due to the poor performance it yielded, we simultaneously experimented with a second approach: taking inspiration from a Demucs-like architecture and attempting to simplify it in a way that made it feasible for us to implement with our available resources

Achieving the same results obtained with the pretrained HDemucs model was impossible due to time constraints and practical limitations, we were unable to perform thorough hyperparameter tuning or conduct long training sessions that might have improved our model's performance. The complexity of Demucs — which includes a hybrid structure operating in both the time and frequency domains, a deep encoder-decoder network, and a BLSTM bottleneck — makes it difficult to replicate from scratch without extensive expertise and infrastructure.

As mentioned before, our goal was simply to build something "artisanal" that could demonstrate the limitations of classical signal processing techniques, without any ambition to match the performance of the most advanced models currently available.

7.1 Implementation

Our implementation centers around a simplified U-Net-inspired encoder-decoder architecture operating entirely in the time domain. The model takes stereo audio input and produces four separated sources: drums, bass, other instruments, and vocals. The architecture consists of several key components that work together to achieve source separation.

The encoder section progressively downsamples the input audio while increasing the number of feature channels. Each encoder block contains a 1D convolution with stride 4 and kernel size 8, followed by channel expansion through a 1×1 convolution and a Gated Linear Unit (GLU) activation. The GLU splits the channels into two parts, applying a sigmoid gate to control information flow, which helps with gradient propagation and selective feature extraction. The encoder reduces temporal resolution from the original 88,200 samples to approximately 1,378 samples at the bottleneck while expanding from 2 stereo channels to 128 feature channels across

four layers.

At the bottleneck, we employ a bidirectional LSTM with two layers to capture long-range temporal dependencies crucial for music source separation. The LSTM processes the encoded features in both forward and backward directions, and its output is reduced back to the original channel count through a 1×1 convolution followed by ReLU activation. This component allows the model to understand musical structure and temporal relationships between different sources.

The decoder mirrors the encoder structure but performs upsampling through transpose convolutions. Each decoder block includes skip connections that concatenate features from the corresponding encoder layer, preserving fine-grained information lost during downsampling. The decoder blocks process features through 3×3 convolutions, channel expansion, GLU activation, and finally transpose convolution for upsampling. The skip connections are crucial for maintaining audio quality and enabling precise source reconstruction.

For training, we developed a combined loss function that operates in both time and frequency domains. The time-domain component uses L1 loss with 85% weight to preserve waveform structure, while the frequency-domain component employs multi-scale STFT loss across different window sizes (512, 1024, and 2048 FFT points) with 15% weight. Additionally, we applied source-specific weighting where drums and bass receive 1.2× importance to address their typically lower separation quality. This multi-scale approach captures both fine spectral details and broader frequency patterns.

Our model integrates seamlessly with the MUSDB18-HQ dataset, properly loading track mixtures and individual stems for training and evaluation. We implemented data augmentation through random gain scaling and channel swapping to improve training robustness. The training process includes gradient clipping, learning rate scheduling, and early stopping based on validation performance.

For evaluation, we implemented standard source separation metrics including Signal-to-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), and Signal-to-Artifacts Ratio (SAR). Our implementation includes robust metric calculation with fallback methods to handle numerical instabilities that can occur with audio processing.

The final model contains approximately 2.8 million parameters and can process audio in real-time on standard hardware. While our results cannot match state-of-the-art models due to the architectural simplifications and limited training scale, the implementation successfully demonstrates functional source separation. The model produces audibly separated sources from mixed audio, with clear isolation of drums, bass, and vocal components, though with some artifacts and cross-contamination typical of simplified architectures.

The key simplifications that enabled our implementation include operating solely in the time domain (avoiding complex frequency-domain processing), using a single bidirectional LSTM instead of more sophisticated recurrent structures, and employing fewer parameters overall. These trade-offs result in a model that is computationally tractable and interpretable while still demonstrating the core principles of deep learning-based source separation.

Our implementation serves its intended educational purpose of illustrating neural

network capabilities compared to classical signal processing methods, providing a practical demonstration of how modern source separation systems operate at a fundamental level, albeit with the expected performance limitations inherent to our simplified approach.

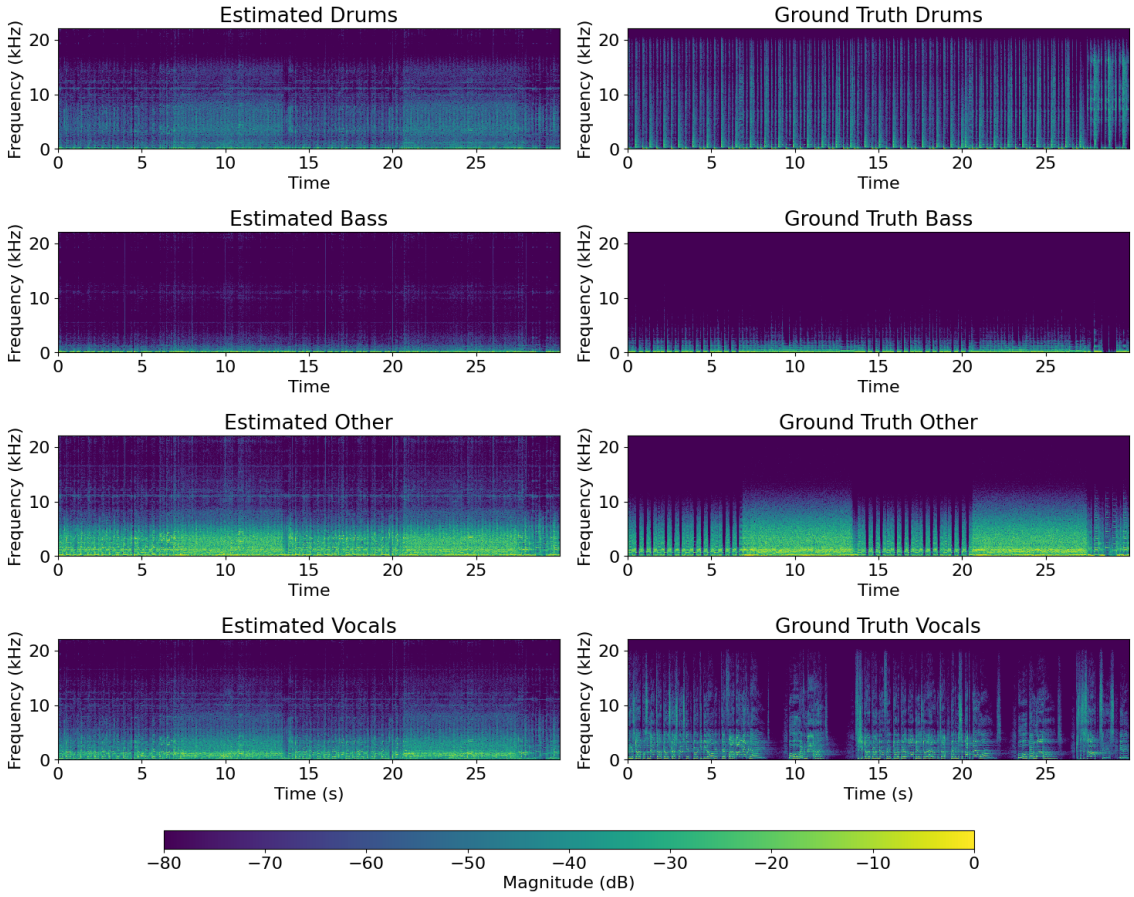

Figure 7.1. Spectrograms Comparison of Simplified Demucs

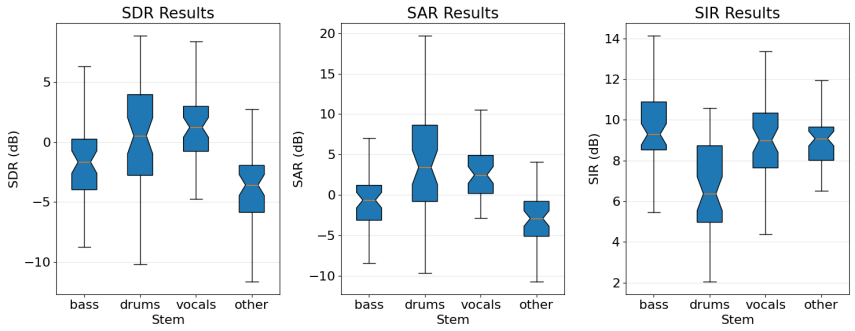

Figure 7.2. Simplified Demucs

Conclusions

After analyzing several techniques for sound source separation—ranging from traditional approaches based on classical signal processing to modern deep learning methods—we explored a wide spectrum of strategies. This included the use of advanced pretrained neural networks as well as the implementation of simpler models developed from scratch. Through this journey, we gained a comprehensive overview of how research in this field is rapidly progressing toward increasingly high-quality results.

We demonstrated how traditional signal processing techniques are now being clearly outperformed by deep learning approaches. In fact, even with limited training, minimal hyperparameter tuning, and relatively simple architectures built by us from the ground up, we were able to surpass the performance of classical methods (not with our U-Net-based model but at least with the simplified Demucs-inspired architecture).

Finally, by testing state-of-the-art models such as Hybrid Demucs (HDemucs), we highlighted how far the technology has advanced—achieving exceptionally clean and precise source separations that would have been unthinkable with conventional methods alone.