POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Evaluation of the impact of the Multi-Head Attention algorithm in Music Source Separation

Supervisors

Prof. Eliana PASTOR

Dr. Moreno LA QUATRA

Dr. Alkis KOUDOUNAS

Candidate

Enrico PORCELLI

April 2024

Abstract

This work focuses on the evaluation of the impact of the Multi-Head Attention algorithm in the field of Music Source Separation. In particular, our objective is to determine its potential to outperform the U-Net architecture often employed in state-of-the-art (SOTA) models. Additional primary goals include examining the repercussions of integrating Self-Supervised features into the pipeline and assessing the efficacy of the Multi-Head Attention mechanism for phase estimation. Notably, when utilizing the magnitude spectrogram as input, our model demonstrated promising outcomes, especially with an increased volume of training data. The incorporation of Self-Supervised features into the model's architecture proved to be effective only when all layer representations are combined into a weighted sum. Blindly concatenating the last layer appeared to be less beneficial to the model's performance. Other findings in this thesis include confirming the utility of the SAD step in the preprocessing pipeline and analyzing the depth of the model, emphasizing once again that Music Source Separation (MSS) models encounter difficulties when the depth is too high. Lastly, it is observed that the Attention mechanism alone is insufficient for accurate phase estimation, a complex task not well suited for the chosen algorithm.

Table of Contents

List of Tables

List of Figures

Acronyms

AAC
Advanced Audio Coding

ADC
Analog to Digital Conversion

AI
Artificial Intelligence

BLSTM
Bidirectional Long Short-Term Memory

CNN
Convolutional Neural Network

DFT
Discrete Fourier Transform

FFT
Fast Fourier Transform

FT
Fourier Transform

FW
Frequency-Wise

GLU
Gated Linear Unit

HT

Hybrid Transformer

ICA

Indipendente Component Analysis

iSTFT

Inverse Short Time Fourier Transform

LL

Lower Limit

LSTM

Long Short-Term Memory

MAE

Mean Absolute Error

MLP

Multi Layer Perceptron

MOS

Mean Opinion Score

MSS

Music Source Separation

NMF

Non-Negative Matrix Factorization

ReLU

Rectified Linear Unit

RMSE

Root Mean Squared Error

RNN

Recurrent Neural Network

SAD

Source Activity Detection

SAR

Signal-to-Artifacts Ratio

SIR

Signal-to-Inference Ratio

SOTA

State Of The Art

SR

Sampling Rate

SSL

Self Supervised Learning

SS

Source Separation

STFT

Short Time Fourier Transform

TW

Time-Wise

UL

Upper Limit

ZCR

Zero Crossing Rate

SDR

Signal to Distortion Ratio

SISDR

Scale Invariant Signal to Distortion Ratio

Chapter 1

Introduction

1.1 The impact of Source Separation

The rapidly evolving landscape of audio and music processing has brought Source Separation to the forefront, proving its significance in technological development and societal impact. SS refers to the task of separating audio sources from a mixed audio signal. Audio sources can be of various nature depending on the application field, for example they can be instruments if we are dealing with Music Source Separation or they could be different human voices and background noises if we are dealing with hearing aids enhancement. Source Separation holds immense potential across a multitude of applications. In the realm of audio and music processing, SS can be used to isolate individual instruments, vocals, or sound elements within a mixed audio track, paving the way for creative remixing and music production. Its utility extends to enhancing speech clarity in noisy environments, enabling better voice communication in hearing aids and telecommunications. Beyond entertainment and communication, SS finds applications in audio analysis, from classifying environmental sounds for ecological monitoring to extracting meaningful features for machine learning-based speech and music recognition systems. It plays a crucial role in audio forensics, aiding in criminal investigations and surveillance. In the medical field, SS assists in isolating and analyzing critical physiological signals, while in industrial settings, it contributes to machinery diagnostics and monitoring. Whether in education, automotive technology, or gaming, Source Separation proves its versatility as an indispensable tool for improving audio quality, enhancing analysis, and enabling innovative solutions across diverse domains.

1.2 Music Source Separation

The task at hand is one of the trickiest ones in the audio processing field and this is mostly due to the simultaneous presence of the different sources, which contribute to the mixed signal with their own timbre, rhythm, loudness and melody. Traditional audio engineering approaches struggle to tackle such a complicated and diversified problem. Trying to untangle these complex sound mixes has often ended up with a leaky or misleading source estimate, where crucial details are missing and information from other sound sources is still polluting the estimate. Source Separation algorithms aim at separating this messy mix of sounds and provide source estimates with greater precision and fidelity to the original source. Music Source Separation (MSS) is a critical enabler for advanced audio manipulation and creative expression. Artists, producers and composers can leverage source-separated tracks to remix and reinterpret existing music. Moreover, the ability to isolate individual sources facilitates the identification and analysis of musical elements, contributing to musicological research and education. From a technological standpoint, Music Source Separation is situated at the intersection of signal processing, machine learning, and audio engineering. As machine learning algorithms and computational power limits continue to advance, MSS techniques are becoming increasingly sophisticated and effective. This evolution presents exciting prospects for real-time applications, enabling interactive music production tools.
In this thesis, we delve into the realm of Music Source Separation, exploring its theoretical underpinnings, algorithmic approaches, and practical applications. By investigating the state-of-the-art (SOTA) methodologies and addressing the challenges inherent to the field, we aim to contribute to the advancement of source separation techniques and their broader implications for music and audio industries. Through empirical evaluations and case studies, we intend to demonstrate the potential of Music Source Separation and highlight the avenues for further research and innovation.

1.3 Problem statement

This thesis addresses the challenge of separating the sources *vocals*, *drums*, *bass*, and the set of remaining sounds, named *other*, from mixed music tracks. The primary objective is to overcome the limitations associated with traditional Music Source Separation architectures, such as Long Short-Term Memory networks (LSTMs) or Recurrent Neural Networks (RNN), which struggle with the vanishing gradient problem and therefore are not optimal for sequential data modelling. Moreover, due to the lack of parallelization capabilities, the aforementioned models are slow during inference and can not be applied to a live stream of data, which is essential

in some applications of Source Separation such as hearing aids development.
In this work we employ various datasets containing tracks from a wide range of musical genres. The thesis revolves around the application of Transformer-based architectures, specifically harnessing the power of Transformer Encoders. The core architecture will output a mask, which will then be applied to the mixture's spectrogram to generate the source estimate.
The aim of this thesis is to contribute to the advancement of the current state of the art in music source separation. By demonstrating the efficacy of Transformer-based models in effectively modeling sequential audio data, the research aims to elevate the accuracy and reliability of source separation outcomes and overcome the traditional U-Net architecture which can be found in several SOTA models.
This study targets researchers as its primary audience, with the potential for broader application in the form of a user-friendly website and with the open-source release of the model weights and architecture. This goal holds the promise of democratizing music source separation and facilitating its integration into music education and production.

1.4 Research Questions

This section contains a list of the questions that we aim to answer with this thesis work. Some of the terms reported below have not been explained yet in order to keep the introduction short. These concepts will be clarified in the background section.

1. Can the limitations associated with vanishing gradient problems and lack of parallelization in traditional Music Source Separation architectures, such as LSTMs and RNNs, be effectively overcome using multi-headed self-attention based models?

2. What are the specific challenges and opportunities in separating *vocals*, *bass*, *drums* and *others* from mixed music tracks, and how can a Transformer-based approach contribute to more accurate source separation?

3. How does the model react to the introduction of additional training data?

4. Can automatically labelled training data be effectively leveraged for expanding the training set and improve performance?

5. How does the proposed architecture perform across a diverse range of musical genres and tracks, and what are plausible underlying factors that contribute to the possibly different performance?

6. What is the impact of selecting the training instances between the active ones using a Source Activity Detector?

7. What is the impact on model's accuracy of the inclusion, during the training phase, of traditional audio features computed for each time frame?

8. What is the impact on model's accuracy of the inclusion of Self-Supervised features computed on the input waveform? And what is the best way to introduce these features into the architecture?

9. Can self-attention effectively deal with the phase estimation task?

1.5 Structure

1. **Background** presents a summary of audio theory fundamentals and a description of the deep learning elements exploited in our architectures.

2. **Related works** contains a brief explanation of the history of MSS algorithms with a focus on the most important research in the field of deep learning-based MSS.

3. **Experiments** contains all the experimental details. These include the description of the experimental settings, the hyperparameter details, a description of the datasets and the explanation of the architectures together with the reasoning behind the choices made. Moreover, in the same chapter we will present the results of the experiments using relevant tables and plots.

4. **Conclusions** is the chapter where we explain and interpret our results. We also propose some ideas for future work on the subject.

5. **Appendix** contains additional material such as mathematical proofs and additional plots and tables that would have impacted the readability of this report if they were introduced in the Experiments section.

1.6 Additional information for the reader

1.6.1 How to read the models' architectures

Following the work in [1] we represent the models designed by us through a Directed Acyclic Graph where edges represent data and nodes represent functions. Black and white nodes identify operations that do not require trainable parameters. A non exhaustive list of this kind of operations includes: softmax, permutations, Short-Time Fourier Transforms, application of pre-trained models for feature extraction

and loss computations. Coloured nodes, on the other hand, represent functions that operate with trainable parameters and these include: weighted sums, linear layers, multi-head self-attention and feed forward neural networks.

1.6.2 Stem notation

In the following chapters we will indicate the target stems with the italic words, e.g. *vocals*. When the term is not italicized, it pertains to the context of the vocals within the song as a whole, without referring to the isolated stem files we are trying to predict.

1.6.3 Reproducibility

To encourage reproducibility of our results, we present the code used for the experiments in https://github.com/enricoporcelli/Music-Source-Separation, together with instructions on how to run it.

Chapter 2

Background

2.1 Audio processing

2.1.1 Audio collection process

Sound is defined as the oscillation in pressure generated by a sound source and propagated through a medium with internal forces (e.g., elastic or viscous), or the superposition of such propagated oscillations [2]. Sound is generated through the vibrational energy produced by vibrating objects. It begins with a disturbance that sets particles, typically air molecules (though they could also be liquid or solid molecules), into motion. When an object vibrates, it causes nearby molecules to compress and rarefy in a repeating pattern, forming sound waves. These waves propagate outward in all directions from the source, carrying the kinetic energy of the vibrations. It's important to note that when a sound is produced, it's not the air molecules that travel; rather, it's the energy originating from the sound source that passes through the air. The air molecules serve as a vibrating medium that transmits the energy.

As the sound waves reach our ears, they cause our eardrums to vibrate in a similar fashion, transmitting these vibrations as electrical signals to our brain. The brain then interprets these signals as distinct auditory sensations, allowing us to perceive and understand the world of sound that surrounds us. Whether it's the strumming of a guitar string, the voice a person nearby or the roar of a car passing next to us, the process of sound transmission and perception is exactly the same.

Recording instruments play a pivotal role in capturing the sounds we emit, enabling us to preserve and analyse them in more details. These instruments, equipped with sensitive transducers such as microphones, function as acoustic sensors that convert variations in air pressure caused by sound waves into electrical signals. When sound waves emanate from sources such as musical instruments, voices, or environmental sounds, they propagate through the air and interact with the

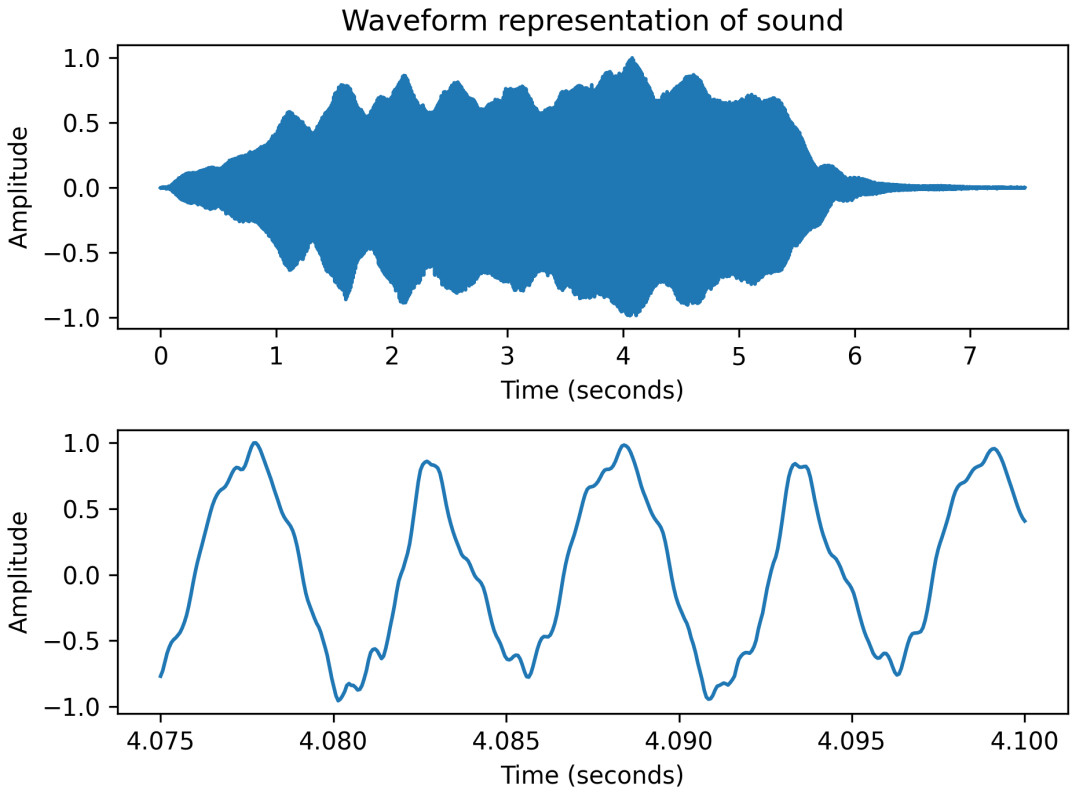

Figure 2.1: Waveform representation of a violin. *Top*: whole waveform. *Bottom*: zoomed version representing 25 ms of audio.

diaphragm of a microphone. This interaction induces minute mechanical movements in the diaphragm, which are then translated into corresponding voltage fluctuations. These are generated by the oscillation of the diaphragm, which is often attached to wires positioned near a magnet. These wires follow the oscillations of the diaphragm and generate the voltage fluctuations accordingly. The amplitude of these fluctuations can be represented against time, thus giving us the so called waveform representation of a sound. An example of this representation can be found in figure 2.1.

2.1.2 The Analog to Digital Conversion (ADC)

Voltage fluctuations, referred to as analog signals, undergo a transformative process into digital representations through analog-to-digital converters (ADCs). The ADC operates in two distinct phases: sampling and quantization. In the sampling phase, equally spaced samples are derived from the continuous-in-time original

signal. The quality of the analog-to-digital conversion is hugely influenced by the number of samples per second (Hz) selected during the sampling phase. Commonly, consumer audio adheres to a standard value of 44.1kHz, whereas archive audio files typically show a significantly higher sampling rate, ranging between 90 and 190kHz. While the higher sampling rate contributes to enhanced sound quality, it poses challenges for mass consumption, primarily because of the larger file sizes generated by this technique. Additionally, the discernible difference in sound quality may not be appreciated by a majority of users. The second phase of the ADC is called quantization and it tackles the problem of continuity in the voltage domain. The quantization phase is where the continuous amplitude values obtained from sampling are mapped to discrete digital values. This transformation involves dividing the range of potential analog values into a finite number of discrete bins, where each bin corresponds to a specific digital value. The crucial factor in this phase is the bit-depth, which defines the number of bits utilized for binary encoding and therefore the number of bins in which the range is divided. Common values include 16, 24 and 32-bit quantizations which yield, respectively, 65.5k, 16.7M and 4.3B levels. The samples obtained before are mapped to the closest bin delimiter through rounding and then encoded in binary form in order to become machine-readable data. If bit-depth and sampling rate are high enough, the resulting digital audio data can capture all the details of the original sound and be indistinguishable from its analog counterpart.
By effectively capturing and preserving these sound waves, recording instruments provide a mean to revisit, study, manipulate and share sound, forming the foundation for various disciplines such as music production, scientific audio research and entertainment. Once the sound is transformed, we can represent the displacement of air molecules over time through a waveform plot. In this visualization, the amplitude of the waveform corresponds to the displacement experienced by the medium's molecules from their equilibrium position. At the same time, the period of the waveform delineates the temporal interval required for an air molecule to complete one oscillatory movement. A fundamental metric in this context, frequency emerges as the reciprocal of the period and is quantified in hertz (Hz), indicating the rate at which these oscillations appear.

2.1.3 The Fourier Transform

Although the waveform representation carries a lot of information such as the amplitude, the loudness, the timbre and other features which are crucial in order to estimate the sound sources, complex waves go beyond the capabilities of the waveform representation alone. An alternative representation is offered by spectrograms, which are a particular type of heat map well suited for audio analysis. To understand how the spectrogram works, we must understand what is a Fourier

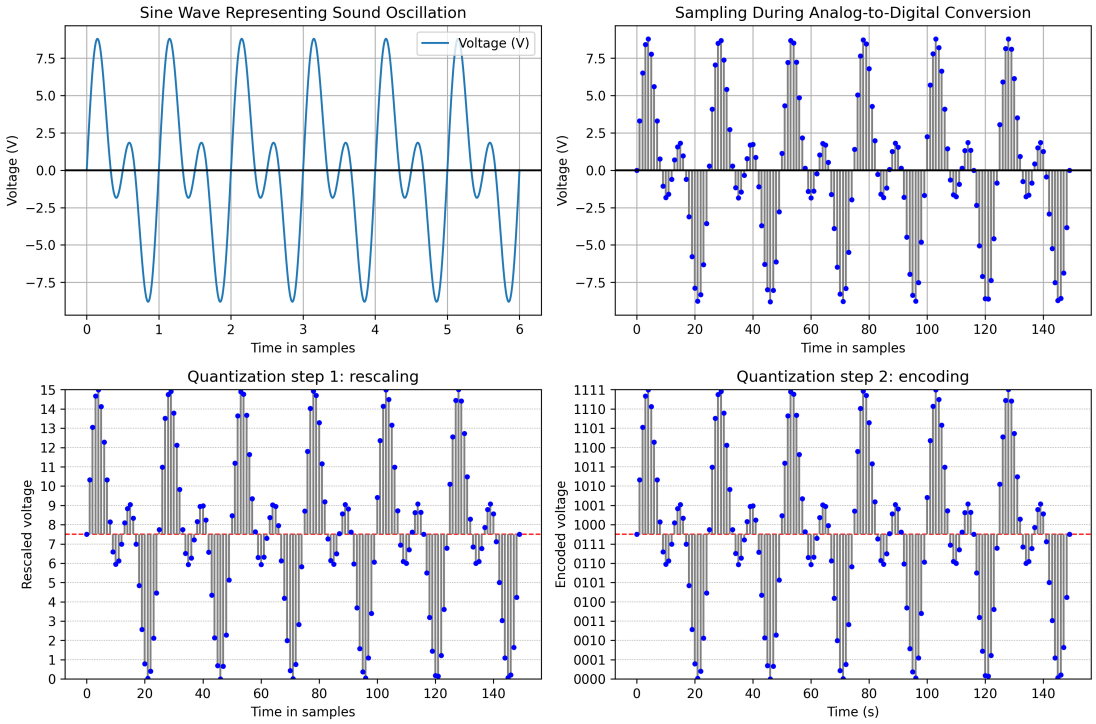

Figure 2.2: Analog to Digital Conversion representation. *Top left*: waveform representation of a sample wave. *Top right*: sampling phase. *Bottom left*: rescaling step of the quantization phase. *Bottom right*: encoding step of the quantization phase.

Transform and what is a spectrum. The Fourier Transform (FT) is a limiting case of Fourier series techniques that deals with the analysis of periodic signals. The Fourier series itself is based on the Fourier's theorem, which states that every periodic function can be expressed as an infinite sum of sine and cosine functions of different frequency and magnitude. Given that each sinusoidal function can be uniquely characterized by its amplitude, phase, and frequency, the process of breaking down the original signal involves determining the phase and amplitude for each potential frequency contributing to the signal. This decomposition allows us to move from the time domain to the frequency domain, where a a sound is measured by its intensity at a given frequency and not by its intensity at a given moment. In the following paragraph I will explain in more details what are the parameters that the FT computes and how it computes them.

The Fourier Series is defined for periodic functions as:

$$f(x) = \sum_{n=0}^{\infty} a_n cos(nx) + \sum_{n=1}^{\infty} b_n sin(nx) \tag{2.1}$$

Where n is the frequency of the sine and cosine and a_n and b_n are the coefficients associated with that specific frequency. By using the sine and cosine complex representations, equation 2.1 can be rewritten as follows:

$$f(x) = a_0 + \sum_{n=1}^{\infty} a_n(\frac{e^{inx} + e^{-inx}}{2}) + \sum_{n=1}^{\infty} b_n(\frac{e^{inx} - e^{-inx}}{2i})$$

Then, by collecting similar terms one obtains:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \frac{(ia_n + ib_n)}{2i} e^{inx} + \sum_{n=1}^{\infty} (\frac{ia_n - ib_n}{2i}) e^{-inx}$$

In the second summation we can now substitute n with -n and change the range of the summation from -1 to $-\infty$ to match the change.

$$f(x) = a_0 + \sum_{n=1}^{\infty} \frac{(ia_n + ib_n)}{2i} e^{inx} + \sum_{n=-\infty}^{-1} (\frac{ia_{-n} - ib_{-n}}{2i}) e^{inx}$$

By doing this, the exponential term is now the same in both summations and we can group the two summations together.

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx} \tag{2.2}$$

Where c_n represents the complex coefficients of the series and is defined so that $c_0 = a_0$. To get a formula for c_n when n is different from 0, we multiply both sides by e^{-imx}:

$$f(x)e^{-imx} = \sum_{n=-\infty}^{\infty} c_n e^{inx} e^{-imx}$$

We can simplify this by integrating both sides from $-\pi$ to π:

$$\int_{-\pi}^{\pi} f(x)e^{-imx} dx = \sum_{n=-\infty}^{\infty} c_n \int_{-\pi}^{\pi} e^{inx} e^{-imx} dx$$

and since the result of the integral $\int_{\pi}^{\pi} e^{inx} e^{-imx} dx$ follows this equation:

$$\int_{-\pi}^{\pi} e^{inx} e^{-imx} dx = \begin{cases} 0, & \text{if } n \neq m \\ 2\pi, & \text{if} n = m \end{cases} \tag{2.3}$$

we obtain:

$$\int_{\pi}^{\pi} f(x)e^{-imx}dx = 2\pi c_m$$

And from this relation we can obtain a closed formula for the complex Fourier coefficient c_n:

$$c_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-inx}dx$$

The complex coefficient obtained through the FT can then be elaborated to extrapolate more information from the audio file such as amplitude and phase of the various frequency components outputted by the algorithm. By writing the FT coefficient as $c_n = a + bi$, where i is the imaginary unit, we can extract the amplitude of the signal of frequency n as $a^2 + b^2$ and the phase as $tan^{-1}(\frac{b}{a})$. Since the series c_{n0}^{∞} is conjugate symmetric then opposite index coefficients will have the same amplitude. A proof of the conjugate symmetric property of the series is reported in the appendix together with a proof of equation 2.3.

The amplitude can be plotted against the corresponding frequency to compose the spectrum, which is a representation of the sound in the frequency domain and can function as an alternative to the waveform representation. Due to the conjugate symmetric property of the series, the spectrogram is symmetrical with respect to the y-axis and for this reason it is often referred to as a double sided spectrum. In figure 2.3 we can see a simplified example of sound composition and the resulting spectrum obtained thanks to the FT.

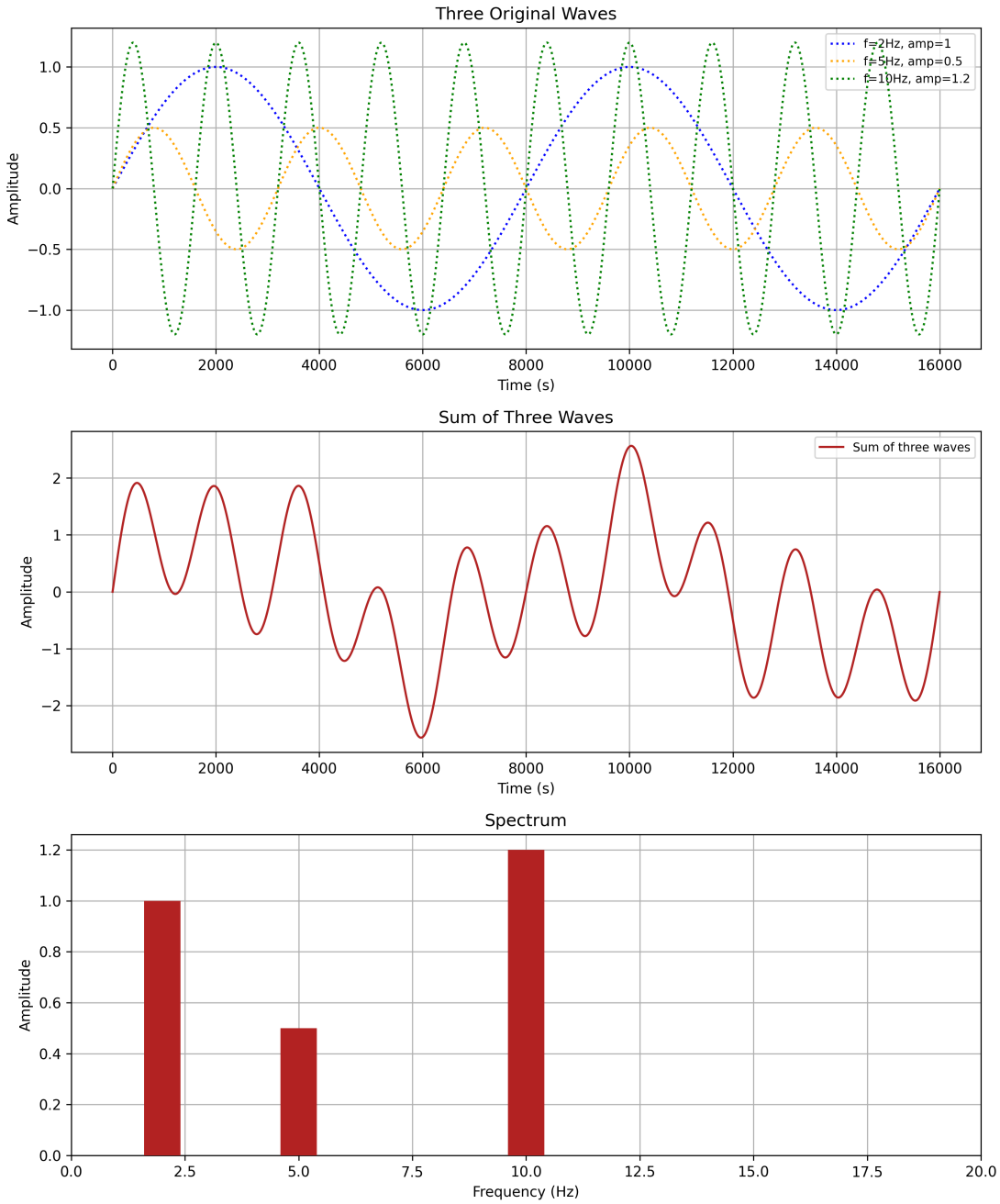

Figure 2.3: Fourier Transform example. Top: three sine waves of different frequency and amplitude. Middle: sum of the three sine waves, representing a simplified real world audio signal. Bottom: resulting magnitude spectrogram obtained through the FT.

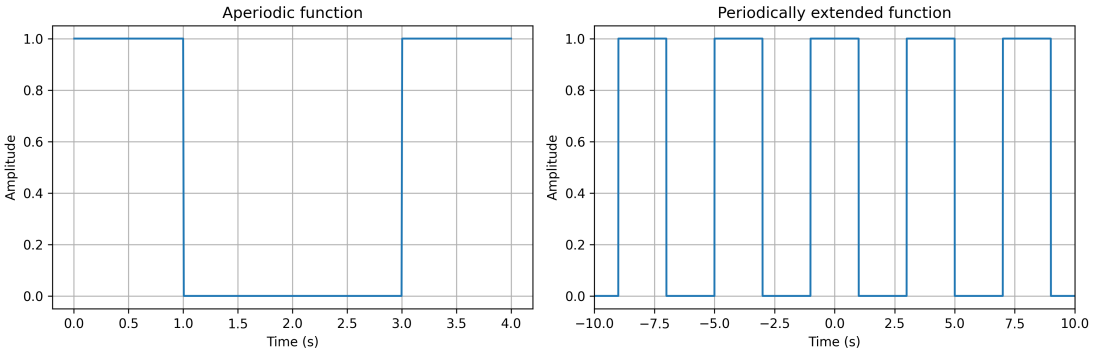

Figure 2.4: Example of periodic extension. On the left: a signal representing a possible recording. On the right: the periodic extension of the signal

There exists a variation of the Fourier Transform (FT) which is adapted to real life signals, which are often aperiodic. In this version, the algorithm periodically extends the signal and then it uses a Fourier series to approximate the periodically extended series. Given an aperiodic signal $x(t)$ and its length T, the periodically extended signal $x_T(t)$ is defined as:

$$x_T(t) = \sum_{k=-\infty}^{\infty} x(t + kT)$$

The original aperiodic signal $x(t)$ can now be expressed as a limit of the periodic extension where T goes to infinity.

$$x(t) = \lim_{T \to \infty} X_T(t)$$

A graphical example of how this process works is offered in figure 2.4

2.1.4 DFT, FFT and The Short-Time-Fourier-Transform

In Computer Science applications, the FT algorithm is approximated with a finite Fourier series through the Discrete Fourier Transform (DFT). This version of the FT can be applied to audio data, which are discrete in time due to the sampling phase of the ADC that we discussed in section 2.2. The issue with DFT is that its computational complexity is $O(N^2)$ for a signal of size N, which is concerning considering the fact that N is often very large in real world applications. The more computationally efficient version of the algorithm is the Fast Fourier Transform (FFT). Developed in 1965, the FFT offers the same results with a complexity of $O(NlogN)$, a reduction in complexity that comes from the so called "divide and conquer" approach. The signal is divided in two, the DFT is computed on both

sub signals and then the result is joined to get the DFT of the original signal. The downside of both the DFT and the FFT is that they are meant to be applied to the whole signal and therefore the resulting spectrum does not contain information on the value of the Fourier coefficients at a given time in the audio, an information that could be useful in many audio processing applications. For example, in the analysis of a song, the FFT would not be able to locate key shifts. With smaller time granularity, one could appreciate a change in the spectrum before and after the key shift, with the amplitude mass moving from frequencies corresponding to the initial key to frequencies corresponding to the new key.
To solve this, one could use an alternative version of the FFT called Short-Time Fourier Transform (STFT) which splits the audio segment in overlapping frames and applies the FFT algorithm to each of them, thus obtaining a different spectrum for every frame. To every frame, a windowing function is multiplied to the digital signal in order to smoothen the bordering samples and avoid discontinuities in the overlapping parts of consecutive frames. The STFT requires the specification of parameters such as the number of samples inside each frame, the overlap between frames and the window size. Although it is usually chosen to be as big as the frame, in certain applications users can choose a bigger window size. Another important factor in determining the output of the STFT is the choice of the windowing function. The most popular choice is by far the Hann function, which is defined as follows:

$$w(k) = 0.5(1 - cos(\frac{2\pi k}{K-1}))$$

with k ranging from 1 to K, where K is the window size. This type of window is sometimes also referred to as the raised cosine window or the Hann filter. In figure 2.6 we can see an example of the application of the Hann window to an audio frame.
The outcome of the application of STFT can be beautifully visualized through a spectrogram. This graphical representation offers a three-dimensional depiction of sound, where time is represented along the horizontal axis, frequency along the vertical axis, and the intensity of each frequency component is reported through a spectrum of colors or grayscale shading. As illustrated in Figure 2.5, spectrograms provide a visual representation that greatly simplifies certain tasks, such as instrument recognition, in comparison to waveform representations. For instance, you can easily spot drum kicks in spectrograms, as they manifest as distinct vertical lines with high intensity across the entire frequency spectrum. Furthermore, spectrograms allow us to discern variations in the intensity of harmonics between bass and vocal sounds. In general, each instrument exhibits a unique distribution of amplitude across its harmonics, and these harmonic patterns can be appreciated through the use of spectrograms.

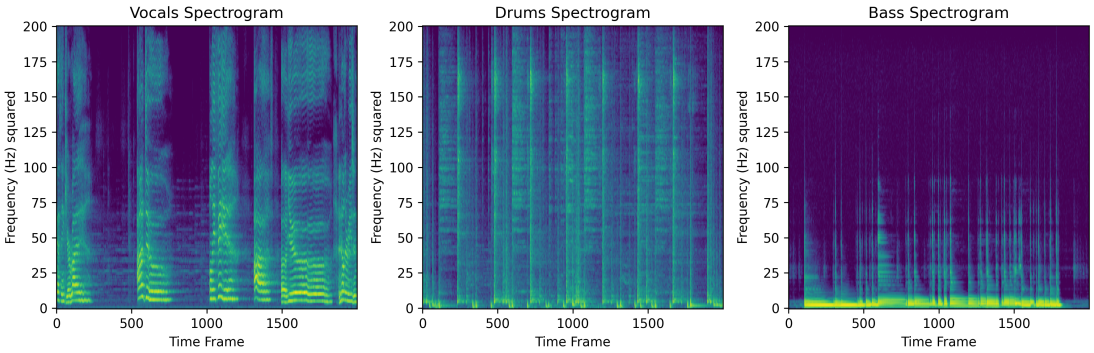

Figure 2.5: Spectrogram example. On the left: spectrogram of a singing voice. In the center: spectrogram of repeated drum kicks sound. On the right: spectrogram of a sound from an electrical bass.

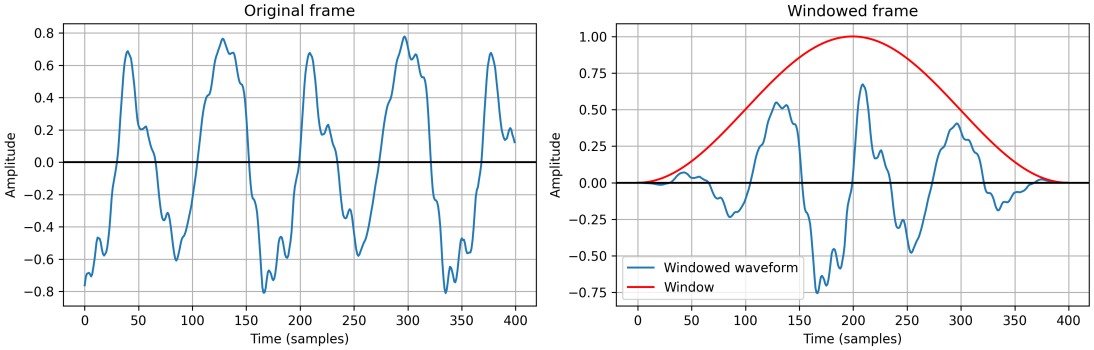

Figure 2.6: Application of the Hann window to a frame. In this example the window size and the frame size coincide and are equal to 400

2.1.5 Limitations of the STFT

Although being the most popular algorithm in signal processing, the STFT comes with a known downside. Due to the Nyquist sampling theorem, the maximum frequency that can be represented without aliasing is equal to:

$$f_N = \frac{SR}{2} \tag{2.4}$$

where SR indicates the sampling rate used in the ADC. Therefore, to avoid aliasing, the STFT is implemented with an upper limit on the frequencies given by the Nyquist frequency expressed in 2.4. During the STFT, the range $[0, f_N]$ is divided

in a number of equally spaced frequency bins which is given by the frame size parameter. In particular, the number of frequency bins will be equal to $\frac{frame\ size}{2}$ and for this reason the frame size parameter of the STFT determines both the time and frequency granularity of the output. Due to this dependency, the main downside of the STFT is having to find the right trade off between frequency and time granularity. Having a higher frame size gives more detailed information about the frequency spectrum but sacrifices details along the time dimension since more samples will be aggregated together in each frame. Conversely, having a small frame size enhances temporal accuracy since less points are aggregated in each frame but this happens at the expense of frequency resolution, because the spectrum information is expressed across fewer frequency bins.

2.2 Audio features

Feature Extraction in Audio Processing involves the extraction of valuable characteristics from an audio file or a spectrogram. Within the wide array of audio features our attention will be directed towards two distinctive waveform domain features. These features can be calculated across the entire waveform or over smaller time frames, providing a different perspective of the audio content. Specifically, we will analyze the Root Mean Square Error and the Zero Crossing Rate, two crucial and commonly used features in the audio processing field. By analyzing these waveform attributes, we can obtain valuable insights into the underlying patterns embedded in the audio signal, thus making audio analysis easier and more complete.

2.2.1 Root Mean Square Error (RMSE)

The RMSE is to quantify the energy content of a signal or a portion of it. It is calculated by taking the square root of the mean of the squared values of a signal's samples. RMSE serves as an effective measure for signal power, enabling the assessment of how much energy is concentrated within a given time frame or frequency band. In audio applications, RMSE can help characterize the intensity or loudness of a sound segment, making it valuable for tasks like audio classification, speech recognition, and quality assessment. High RMSE values generally indicate higher energy or louder portions within an audio signal, while lower values suggest quieter or less energetic segments. This metric is particularly useful for identifying and distinguishing sound events. Here is the formula to compute the RMSE in a particular time window:

$$RMSE(t) = \sqrt{\frac{1}{n} \sum_{n=N_0(t)}^{N_T(t)} x_n^2} \tag{2.5}$$

where $N_0(t)$ is the vector containing the sample indexes of the starting sample of every frame. Similarly, $N_T(t)$ contains the indexes of the ending samples of every frame.

2.2.2 Zero Crossing Rate (ZCR)

The Zero-Crossing Rate (ZCR) is a crucial audio feature used in a wide range of audio and speech processing tasks. It quantifies how rapidly a signal changes its polarity, effectively counting the number of times the signal crosses the zero amplitude point within a specified time frame. ZCR serves as a valuable indicator of the noisiness of an audio segment, providing insights into characteristics like percussiveness in music, fricatives in speech, or even the presence of discontinuities in sound. In music analysis, for instance, high ZCR values may indicate a highly percussive sound or rhythm, while low ZCR values may correspond to more tonal or sustained components. In speech recognition, ZCR can help identify voiced and unvoiced segments, aiding in phoneme and feature extraction. This feature plays a significant role in distinguishing and characterizing different sound types, making it a key component in audio classification, speech processing, and even music genre recognition. The reason why the zero crossing rate is so high in percussive sounds is because percussions generate sounds through the vibration of membranes. Membrane vibrations can generate sounds with frequencies ranging from 100Hz up to the limit of human hearing (around 20kHz). These higher frequency sounds result in more crossings of the zero-amplitude line when compared with melodic instruments, where higher frequency are only reached by the harmonics, which are often lower in magnitude with respect to the tonic and therefore have lower impact on the sound wave shape. In figure 2.7, we showed the results of an exploratory analysis performed on the MUSDB18 dataset[3] which shows that the average ZCR across all frames in a song can help distinguish between the bass, voice and drums sources. Defining N_0 and N_T as we did before, we can write the formula for the ZCR of a frame t as:

$$ZCR(t) = \frac{1}{N_T(t)} \sum_{n=N_0(t)}^{N_T(t)} 1_{[\leq 0]}(x_n x_{n-1})$$

The ZCR ranges from 0 to 1 and expresses the ratio between the crossings of the zero line and the number of samples in the frame analysed.

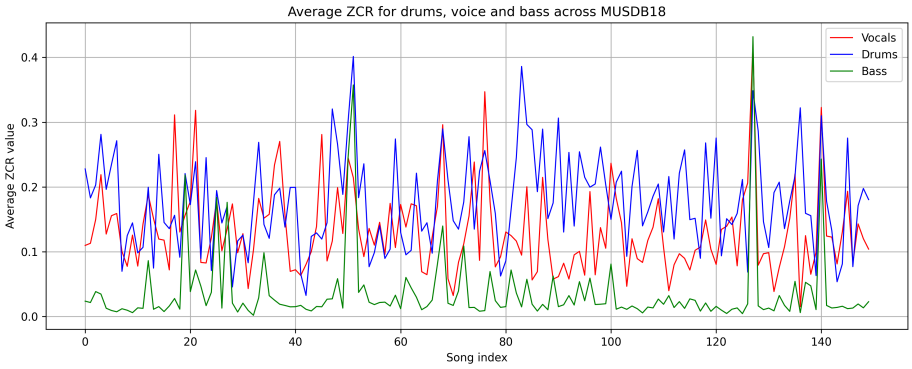

Figure 2.7: Average frame-level ZCR value computed for every song in the MUSDB18 dataset[3] for vocals, drums and bass sources. The frames are selected with frame size equal to 4096 and hop length equal to 2048. The average frame-level ZCRs of across the whole dataset for the different sources are $ZCR_{drums} = 0.181$, $ZCR_{vocals} = 0.129$, $ZCR_{bass} = 0.036$

2.3 Deep-learning

2.3.1 Multi Layer Perceptrons

A Multi Layer Perceptron (MLP) [4] is a type of artificial neural network that consists of multiple layers of interconnected computational nodes. In an MLP, information moves through the network in a feed forward manner, with each layer processing the output of the previous layer and transforming it using weighted connections and activation functions. The network typically comprises an input layer, one or more hidden layers, and an output layer. In every neuron in the network, a weighted sum of the inputs is calculated, followed by the application of an activation function to produce the neuron's output. This process can be mathematically represented as follows: let x be the input vector containing n elements, $W^{(i)}$ be the weight matrix for layer i, $b^{(i)}$ be the bias vector for layer i, $z^{(i)}$ be the weighted sum at layer i, $a^{(i)}$ be the output of layer i after applying an activation function $f^{(i)}$.
For a hidden layer i, the weighted sum $z^{(i)}$ is calculated as:

$$z^{(i)} = W^{(i)}a^{(i-1)} + b^{(i)}$$

The output $a^{(i)}$ is obtained by applying an activation function $f^{(i)}$ to the weighted sum:

$$a^{(i)} = f^{(i)}z^{(i)}$$

With common choices of $f^{(i)}$ being the ReLU, Tanh or Sigmoid functions. This process continues through the network until the final output layer is reached, providing the predicted output for the given input.
What sets MLPs apart is their ability to potentially model any complex, nonlinear relationships within data [5], making them well-suited for a wide array of machine learning tasks, from regression and classification [6] to, more recently, image classification [7]. These networks can be trained using various algorithms, most notably backpropagation, to minimize the difference between predicted and actual output, iteratively adjusting the weights associated with each connection [8].
The architecture and size of MLPs can be tailored to the specific requirements of a task, with larger networks capable of capturing more intricate patterns but also demanding more data and computational resources. MLPs have been instrumental in the development of deep learning, serving as the foundation for deep neural networks and they remain a cornerstone of modern machine learning and although in the past they were mainly used as classifiers with outstanding success, today they find a purpose in many complex architectures as a tool for extracting features or reshaping input vectors [9]. For this last case, MLPs without hidden layers are often the best choice and such MLPs are referred to as Linear Layers or Fully-connected layers.

2.3.2 Convolutional Neural Networks

Convolutional Neural Networks, often abbreviated as CNNs [10], are a class of deep learning models that have revolutionized the field of computer vision and, to a growing extent, other domains. Inspired by the visual processing of the human brain [11][12], CNNs are designed to automatically and adaptively learn hierarchical patterns and features from data. They excel at tasks involving grid-structured data [10], with their distinctive feature being the use of convolutional layers. These layers apply a set of learnable filters or kernels to input data, typically two-dimensional arrays like images, which slide over the input, computing element-wise multiplications and aggregating results to detect local patterns, edges, textures, and more complex features [10].
Indicating with x the input and with w the filter, the feature map z is computed as:

$$z = f(x * w + b)$$

Where f is an activation function to introduce non linearity. This iterative process makes CNNs capable of understanding the local context within an image. Moreover, CNNs often include pooling layers that downsample the feature maps, thus reducing the computational complexity of the algorithm by aggregating regions of data. With the extracted features, CNNs can then be concatenated with one or more fully

connected layers to perform high-level reasoning and make predictions [13] [1]. CNNs are employed in a wide array of applications beyond image recognition. Notably, deep CNN architectures like the VGG, ResNet, and Inception networks have demonstrated remarkable performance in image classification in object detection tasks [13][1][14], further emphasizing the versatility of CNNs.

2.3.3 LSTMs

Long Short-Term Memory, or LSTM [15], is a specialized type of recurrent neural network (RNN) that has made a profound impact on sequential data processing tasks. LSTMs are designed to address a fundamental issue in traditional RNNs, namely the vanishing gradient problem, which can hinder their ability to capture long-range dependencies in sequences [15]. LSTMs achieve this by introducing a more sophisticated memory cell that can store information over extended time intervals. This memory cell has three key components: an input gate, a forget gate [16], and an output gate. The input gate determines what new information to store in the cell, the forget gate regulates what old information should be discarded, and the output gate controls what information should be used to make predictions or be passed on to the next time step. This architecture allows LSTMs to effectively handle sequences with gaps between important events or with a need to store and retrieve information from the distant past.
LSTMs have found extensive use in various applications, including natural language processing [17], speech recognition [18] [19], machine translation [20] and, more generally, in tasks where capturing context and handling temporal dependencies are crucial. Furthermore, variations and extensions of LSTMs, such as Gated Recurrent Units (GRUs) [21] and bidirectional LSTMs [22], have emerged to cater to specific requirements and further improve the network's capabilities. While LSTMs are powerful, their biggest drawback is that due to their recursive nature they are not parallelizable and therefore the training phase is slower with respect to a traditional feed forward network. Despite this, they remain one of the most valuable tools for sequence modelling in the deep learning field.

2.3.4 Transformers

The Transformers architecture is a groundbreaking innovation in deep learning and natural language processing, which has revolutionized the way we approach sequence-to-sequence tasks and significantly improved the state of the art in various AI domains. Developed by Vaswani et al. in the paper "Attention Is All You Need" [23], Transformers are designed to handle sequential data efficiently through a mechanism called self-attention. Unlike traditional recurrent or convolutional networks, Transformers process input data in parallel rather than sequentially,

making them highly parallelizable and thus compatible with efficient training on modern hardware. Central to the Transformers' success is their implementation of the attention mechanism [24], which allows each element in a sequence to consider the entire context when making predictions, thereby capturing long-range dependencies effectively. Mathematically speaking, the attention mechanism applied to a single input vector is a weighted sum of all the vectors in the sequence. The result will be an embedding of the input vector which will be based on the context, i.e. the other vectors in the input sequence.

$$e_i = \sum_{n=1}^{N} \alpha_{in} v_n$$

The key novelty in the paper is the how the weights α_{in} are computed: for every vector, three different linear transformations of its embedding will constitute its query, key and value vectors which are all of the same length. The weight α_{in} is computed starting from the vector multiplication between the query vector of the input vector i and the key vector for the input vector n. The result is called attention score and is computed for every element in the input sequence to obtain a vector of attention scores and they define a relation between elements of the input sequence. These are set to have sum equal to 1 through a softmax operation. The resulting weights are then multiplied by the value vectors to obtain the elements needed to compute the sum which defines the embeddings. We can denote with X the matrix containing the input sequence and with W^Q, W^K and W^V the linear transformations needed to compute the queries, keys and values, which will be collected in the matrices Q, K and V. We can now express the attention mechanism in a compact way through the following definition:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where d_k is the dimensionality of the key vectors, which in the original Transformer implementation is the same as the dimensionality of the query and value vectors. This process has the big advantage of being parallelizable and of allowing a vector representation to be aware of the entire context of the input sequence, thus effectively contrasting the vanishing gradient problem encountered in previous SOTA models for sequence modelling such as RNNs and LSTMs. A more sophisticated attention mechanism has been proposed in the original Transformer paper [23] and is called Multi-Head attention and the output embeddings are given by:

$$M - HAttention = Concat(head_0, ..., head_i, ..., head_h)W^O$$

where:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

and W^O is a linear transformation which has the role of combining the attention heads results and reshaping the concatenated vector. The matrices W_i^Q, W_i^K, W_i^V are the 3 different matrices used to obtain queries, keys and values respectively for $head_i$. Practically speaking, this is equivalent to the repetition of the attention mechanism h times, thus encoding the same vector based on the entire input sequence in h different ways and then combining those h encodings through a linear transformation W^O.
The architecture consists of an encoder and a decoder, each composed of 6 layers. In the encoder, the previously explained self-attention mechanisms is followed by a feed forward network which has the role of processing the result of the attention mechanism in order to make it optimal for the one in the following layer. The decoder, on the other hand, first generates an embedding of the input sequence using masked self-attention, which is a self-attention mechanism computed only on preceding input vectors to avoid predictions based on future elements of the sequence. After that, the contextualized output embedding is used to compute the query in the next attention mechanism, which is often referred to as the encoder-decoder attention. Here the keys and values are computed with the linear transformations $K = W^K X_E$, $V = W^V X_E$ starting from X_E which is the matrix containing the embeddings obtained after the 6 encoding layers. The query is computed with the usual linear transformation $Q = W^Q X_D$ but in this case it is a linear transformation of X_D, which is the output of the masked self attention of the current decoder layer. Transformers have become the foundation of many state-of-the-art natural language processing models, including BERT [25] and GPT-3[26], which have achieved remarkable results in tasks like language understanding, translation, summarization, and question-answering. Although the Transformer architecture requires bigger training datasets compared to other deep learning models, it has probably been the most impactful discovery in the field of deep learning since the invention of CNNs.
In this thesis, we will assess the effect that the attention mechanism and the transformer encoder have in the field of Audio Source Separation.

2.3.5 Self-Supervised Learning

Self-Supervised Learning (SSL) represents a framework in deep learning where a model is trained on a dataset that does not contain labeled target information. Unlike traditional supervised learning, where input-output pairs guide the training process, SSL exploits structures within the input data to generate supervisory signals without the need for data labelling. This framework works through the designing of an auxiliary task that exploits the structure of the training samples. Different

research fields require different auxiliary tasks in order to train a model that is able to generate meaningful representations of the data. SSL has demonstrated to be successfull in various domains ranging from natural language processing [25][26] to speech processing[27] and is particularly popular for general purpose pre-trained models, where a large quantity of data is required. These models are typically used for generating embeddings for various downstream tasks. Famous examples in the audio domain are presented in the original papers, where the embeddings have been used for speech recognition [27][28][29], speaker verification, speaker diarization and speech separation [28].

Chapter 3

Related Works

This section serves as an overview on the current status of Music Source Separation algorithms and as a tool to have a better understanding of the experiments that will be performed in this thesis. With the first section titled Traditional Approaches to Source Separation we will see which are the precursors of modern algorithms in order to have a better perspective of the path that this research field is following. It must be noted that, while all the methods discussed are applicable to Music Source Separation, most of the papers cited in this first section focus on speech separation, speech enhancement or very simple MSS scenarios since the field of SS was mainly focused on speech tasks in the early years of its developing. Speech algorithms can still be useful for MSS, an example in today's SOTA models is given in [30], where state of the art speech separation algorithms from Ultimate Vocal Remover [31] are used to isolate the vocals of a song and then subtracted to obtain an audio without the vocals, thus making the MSS task easier.
The second section is centered on modern deep learning algorithms, which can be classified in three categories: time domain, frequency domain and mixed domain algorithms. The third section focuses on the Conformer algorithm for speech-related tasks, which will turn out to be useful in our experiments. In the last section the reader will find an overview of common evaluation metrics used in the literature, together with a discussion about their strength and limitations.

3.1 Traditional Approaches to Source Separation

Before the advent of deep learning, Source Separation algorithms relied on statistical models with few parameters and linear algebra techniques to separate the signal into its components. In the following subsections we will take a look at the most popular ones, namely Independent Component Analysis, Non-Negative Matrix Factorization and Spectral Clustering.

3.1.1 Independent Component Analysis (ICA)

ICA for Source Separation was a popular method that aims to linearly combine data into components which have the characteristic of being as statistically independent as possible. These linear combinations of the observed signals should also display an *interesting* distribution, where the word *interesting* indicates a distribution that displays some structure. Referring to the arguments reported in [32][33], the authors of the paper consider the Gaussian distribution as the least interesting one. In ICA, the problem of Source Separation is formulated as a latent variable model estimation where the components are chosen with the objective of maximizing the non-Gaussianity of the components while keeping them as statistically independent as possible[34]. In this setting, both the sources and the mixed signal are considered as random variables while the recordings of this signals are considered as realizations of these random variables. ICA has been proposed in [34] where it has been effectively used in audio processing tasks such as blind source separation and feature extraction. In the same paper, an efficient algorithm called FastICA has been proposed for the estimation of the sources.

3.1.2 Non-Negative Matrix Factorization (NMF)

NMF is a matrix factorization technique that decomposes a matrix into a set of basis vectors and their associated activations, with the constraint that all elements in the factorized matrices are non-negative[35]. While in the original paper it has been applied to images, the translation to the audio processing field is trivial if we use spectrograms. The algorithm is based on the idea, confirmed by psychological studies[36][37], that the perception of the whole is based on the perception of its parts. NMF revolves around the approximation of a matrix as the product of two non-negative matrices with the aim of minimizing the reconstruction of the original matrix. One matrix will contain the bases of the signal as its columns, while the other one will contain the weights of the basis as its rows. The different signals can then be reconstructed by multiplying one basis by the corresponding weights. In its base form, NMF applied to audio processing dicards the time information of the spectrogram but of great interest is the work of Smaragdis [38], who developed an algorithm that is able to take into account the sequentiality of the spectrums in the spectrogram produced by the STFT. Other successful applications of NMF in the context of audio source separation can be appreciated in the works [39][40].

3.1.3 Spectral Clustering

Spectral clustering is an unsupervised source separation technique that first decomposes the audio signal into its sinusoidal modelling representation and then groups frames into cluster and separate sources based on their spectral characteristics[41].

The spectral characteristics considered are the amplitude, phase and frequency and the technique used for clustering. To perform the clustering, the first step is to generate a graph for the audio file where every node is a frame and the edge between frames is given by the similarity between them. Similarity is expressed by mean of various measures of proximity between the frequencies, amplitude and harmonicity of the two frames connected by the edge [42][43]. Once the graph has been computed and the weight matrix has been defined, the Normalized cut algorithm [43] is applied in order to maximize the similarity among elements of the same cluster and maximize the average dissimilarity between elements of different clusters. Recent work on the subject [44] resulted in a faster implementation of the algorithm to compute the similarity matrix, making it 30x faster.

3.2 Usage of RMSE and ZCR in literature

ZCR and RMSE have been used in Source Separation literature as an additional source of information to other waveform and cepstrum features [45][46]. Although they cannot be used as the primary source of information, figure 2.7 suggests that, for a Music Source Separation algorithm, the ZCR could be helpful in identifying drums and allow the algorithm to either either isolate them or remove them in case the objective is the estimation other instruments. ZCR and RMSE have also shown good potential in the identification of voiced and unvoiced sections of audio [47], which can turn useful for vocals track estimation and, similarly to the previous example, for the estimation of the other tracks through removal of the estimated source from the mixture.

3.3 Relevant Deep Learning Algorithms

3.3.1 Deep Learning in the frequency domain

Deep learning algorithms for MSS in the frequency domain started to gain popularity thanks to the explosion of computer vision. This is because the spectrogram offers a direct image representation of a song and therefore all the algorithms that have found success in the image processing field could be easily translated in the audio processing one using the frequency domain representation of audio. The success that the U-Net architecture achieved in biomedical image segmentation[48] led to a growing interest in the architecture and when applying the U-Net algorithm to audio processing, the simplest idea has been to repeat the same process on spectrograms. With this similarity in mind, a combined effort from researchers at the University of London and at Spotify delivered a successful algorithm for singing voice separation[49]. Similarly to the original U-Net, this version implements 6

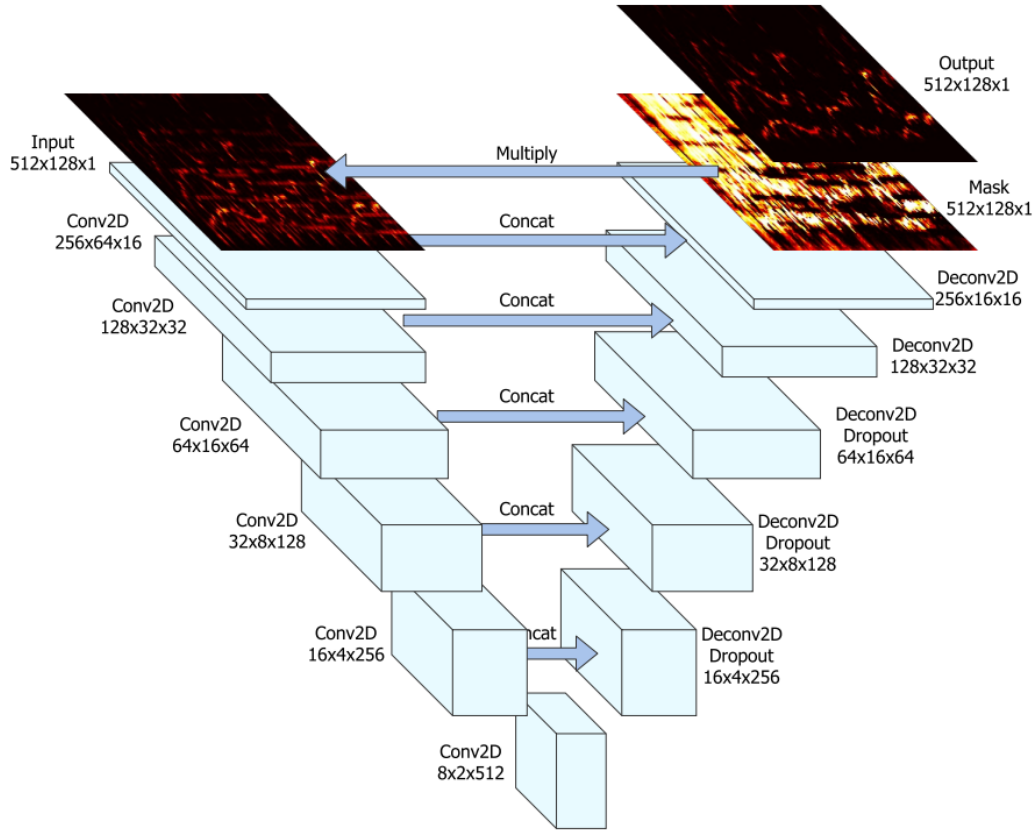

Figure 3.1: U-Net architecture applied to the magnitude spectrogram. Image from the original paper [49]

encoding layers followed by 6 decoding layers and connects each level of layers with a skipped connection. In the encoding branch, 5x5 convolutions are applied to the magnitude spectrograms and the resulting feature is then upsampled through six deconvolution layers which compose the decoder branch. The result of this process is a map, which is then multiplied to the original magnitude spectrogram to obtain an estimate of the singing voice track. The original representation of the architecture is shown in figure 3.1

While it marked a distinct step ahead in the research field by obtaining top results on most metrics, the ability to separate only the vocals stem made it more similar to a Voice Detection algorithm applied to music data rather than an actual Music Source Separation algorithm. Later in the same year (2017) researchers at Sony released the first open-source SOTA model, called Open-Unmix[50] and based on three bidirectional LSTMs applied to the magnitude spectrogram. The same architecture has been trained four times on the different MUSDB18 sources, namely

vocals, drums, bass and others, to generate a set of weights for each source. From then onward algorithms working on the magnitude spectrogram have been at the basis of most SOTA architectures, an example is D3Net[51] which achieved an average median SDR of 6.0dB across the 4 stems on the MUSDB18 dataset[3] in 2021 by combining dense skip connectivity with dilated convolutions to broaden the receptive field of the CNNs and facilitate the flow of information between layers. All these models work on the magnitude spectrogram, with all the limitations that neglecting the imaginary part of the spectrogram can present. With more recent works [52], these limits have been surpassed using the full complex spectrogram as the input for the models. A way to treat the imaginary part is through the so called complex-as-channel[53] method, where the real and imaginary parts of a complex valued spectrogram are concatenated to form the input of the model. A second way of treating complex valued spectrograms is through MLPs which encode the real and imaginary parts of a spectrogram into real valued representations. This is the approach utilized by the inventors of the Band-Split RNN [52], which is currently the best performing model among the spectrogram based ones with an average median SDR of 8.23dB on MUSDB18. The model features a novel way of managing the complex spectrogram in which the frequency bands produced by the STFT are grouped together and passed through different MLPs to produce a feature for each group. In lower frequencies, fewer frequency bands are grouped together, whereas in the higher frequency range more bands are grouped as less information is contained in those ranges. The features are then passed twice through a a Time-Wise BLSTM, a Frequency-Wise BLSTM and a fully-connected layer. The resulting features are then mapped back to spectrogram form, leaving to one different MLP per group the task of dealing with the Real-to-Complex transformation. There exists a version of the model which uses an additional dataset for training in addition to the standard MUSDB18 training set. This version reached an average median SDR of 8.97dB.

3.3.2 Deep Learning in the waveform domain

As highlighted in the introduction of [54], the managing of the phase information is a divisive topic in the field of Music Source Separation. For many years, most researchers relied on magnitude or power spectrogram to make computations easier and faster, thus completely ignoring the phase information. At the end of the decade going from 2010 to 2020, researchers have started to question this approach, wandering whether ignoring the phase information would hinder their possibility of developing a reliable source estimate from the mixed signal. During that decade, the increase in computational power and the introduction of new deep learning algorithms led the MSS research field to a newly found interest in waveform domain solutions, which do not require the distinction between magnitude and phase and work with the complete information retrievable from the audio file. MSS in the

waveform domain saw its first breakthrough with the Wave-U-Net model [55] in 2018. As the name suggests, it is inspired by the success of U-Net and it follows the same encoding-decoding structure but substituting 2-D Convolutions with 1-D ones in order to adapt it to time domain audio data. After Wave-U-Net, in the same year came another relevant waveform domain MSS model called Conv-Tas-Net. Here the input waveform is divided into overlapping frames, transformed in a N-dimensional vector by means of a 1-D convolution and then passed through several layers composed of a set of stacked 1-D dilated convolutions, whose output is zero padded and passed to the following layer of identically stacked convolutions. The output of each layer is then used to compute a mask for the target source, which is applied to the N-dimensional representation of the audio. Reportedly inspired by the success of U-Net architectures in music synthesis [56], researchers at Facebook developed the model from which the current State Of The Art architecture is inspired. The model is called Demucs [57] and is composed of six encoding layers, two BLSTM layers and six decoding layers. Following the Wave-U-Net architecture, every encoding layer is based on convolutions and its output is sent to the next layer and to the corresponding decoding layer through a skipped connection to make the flow of information easier through the network. Key differences with Wave-U-Net include the choice of the GLU activation functions in the encoder and decoder layers and the insertion of the two bidirectional LSTMs before the decoding branch begins. Demucs has been trained with additional training data and its performance has stayed at the top of the MUSDB18 leaderboard from 2019 to 2021, when hybrid networks set new SOTA standards.

3.3.3 Deep learning in mixed domain

A lot of the latest algorithms for MSS try to use information from both the time and frequency domains. The first successfull algorithm to exploit this idea was the KUIELab-MDX-Net [58], which used two different branches for the two domains and only at the end it merged their results to create a final estimate of the target source. In the same year, researchers at Meta AI released Hybrid Demucs, a LSTM based architecture that follows the encoding-processing-decoding structure of the original Demucs. The difference here is that there are two branches in order to manage both the waveform and the spectrogram information. It differentiates from the KUIELab-MDX-Net because before the processing phase, the two branches are concatenated in order to allow for cross-domain processing. This means that the waveform branch can capitalise from the information received from the spectrogram branch and viceversa. This architecture has been on top of the MUSDB18 leaderboard from 2021 to 2022, when the Band-Split RNN architecture claimed the fist place. During the development of this thesis, the Demucs architecture has been upgraded and a new version called Hybrid Transformer Demucs (HT Demucs)

[59] was released. As the name suggests, this is an evolution of the Hybrid Demucs architectures that implements self-attention and cross-attention mechanisms during the processing phase, which makes the flow of information across domains easier and more successfull. This architecture achieved an average median SDR of 8.80dB across the 4 stems on MUSDB18 with the help of additional training data and it claimed the top of the leaderboard with an average median SDR of 9.20dB thanks to per source fine-tuning and to the addition of sparse attention kernels in order to amplify the receptive field of the attention mechanism.

3.3.4 Useful deep learning algorithms from related research fields

In this section we will delve into the Conformer algorithm [60] and, in particular, into its implementation in [61]. The Conformer uses a structure similar to the Transformer Encoder but it inserts a convolution module between the multi-head self-attention and the feed-forward blocks. The convolution module presents, together with normalizations and activations, 2 pointwise convolutions at the beginning and at the end and a 1D-depthwise convolution in the middle of the module. The application of this architecture to the field of Music Audio Enhancement has been extensively studied in [61]. Of particular interest are the first two implementations of the Conformers. The first one uses a first time-wise Conformer on the spectrogram to then pass the output to a second frequency-wise Conformer block and is referred to as the cascade implementation. A second version of the algorithm divides the two Conformers into separate branches and is referred to as the parallel implementation. These two architectures will inspire some of the architectures used during the experimental phase of this thesis.

3.3.5 Self-Supervised Features

In recent years, the success that Self Supervised Learning (SSL) found in NLP [25] led to a growing interest in the capabilities of Self-Supervised general pre-trained audio models and several efforts have been done to train models that were as versatile as possible. Famous examples of these models include HuBERT [29] and Wav2Vec[27], which produce quality features but have been trained on a dataset mainly composed of audiobooks. This thesis will utilize audio features produced with a general pre-trained audio model named WavLM [28], which uses masked speech denoising and prediction as the training objective and exploits a modified Transformer Encoder as the backbone together with several convolutions to better encode the input of the transformer. WavLM Base+ includes 12 Transformer Encoder layers which output features of dimension 768. The larger version of the model is called WavLM Large and includes 24 Transformer Encoder layers with

feature dimension equal to 1024. Although the use of WavLM Large yields greater improvements in many downstream tasks[28], to ease the computational effort this thesis will implement WavLM Base+ for the production of complementary audio features from the waveform. We predict these features to have a greater impact in the *vocals* predictions since the model has been trained on speech data. In a follow-up experiment we will also test the impact of newly trained versions of HuBERT Base, HuBERT Large, Wav2Vec2 Base and Wav2Vec2 Large. The new model versions have been developed by ALM research team and use a much more diverse training dataset called Audioset, which is composed of multiple music and speech audio datasets. The features generated by these models achieve SOTA performances on several audio classification tasks[62] and with this experiment we aim to verify whether these results translate to MSS too.

3.4 Common evaluation metrics

This section will be dedicated to the description of the three most common evaluation metrics in Music Source Separation, namely the Signal-to-Distortion Ratio, the Signal-to-Artifacts Ratio and the Signal-to-Inference Ratio. Together with their description, we will also highlight their strength and weaknesses. The introductory section focuses on the work related to error decomposition in Source Separation, which is the foundation for the three metrics described in the following sections.

3.4.1 Error decomposition

Music Source Separation metrics rely on the error decomposition described in [63]. Every metric relies on the assumption that a source estimate is composed as follows:

$$\hat{s_i} = s_i + e_{noise} + e_{interference} + e_{artifacts} \tag{3.1}$$

where s_i is the ground truth and the other terms represent the error terms for noise, interference and artifacts. More details on how to calculate e_{noise}, $e_{interference}$ and $e_{artifcats}$ can be found in [63] together with more details regarding their derivation.

3.4.2 Signal-to-Artifacts Ratio (SAR)

The SAR computes the amount of artifacts in the estimate, with an artifact being a sound that is not present in the target source nor in the other sources.

$$SAR = 10\log_{10}(\frac{||s_{target} + e_{interference} + e_{noise}||^2}{||e_{artifacts}||^2})$$

3.4.3 Signal-to-Inference Ratio (SIR)

The SIR measures the amount of leaking present in the estimate. In Audio Processing, leaking refers to the situation in which the source estimate contains sounds from other sources

$$SIR = 10\log_{10}(\frac{||s_{target}||^2}{||e_{interference}||^2})$$

3.4.4 Signal-to-Distortion Ratio (SDR) and SI-SDR

Also called Signal-to-Noise Ratio (SNR), the SDR is computed as the ratio between the ground truth and the total amount of distortion in the estimate, i.e. the sum of the three errors in equation 3.1. For a wave of length N, the formula for the SDR is defined as:

$$SDR = 10\log_{10}\frac{\sum_n^N ||s(n)||^2 + \epsilon}{\sum_n^N ||s(n) - \hat{s(n)}||^2 + \epsilon}$$

where ϵ is a small constant which is set to 10^{-7} with the purpose of avoiding divisions by 0. The SDR is measured in dB and current state of the art models are able to achieve an average median SDRs of 9.2 across the four stems in MUSDB18. If the quantity of the captured signal is equal to the amount of captured noise, the SDR will have value equal to 0dB, if the noise in the estimate is less than the signal, then the SDR will increase. The range of the SDR is bounded between $-\infty$ and the following Upper Limit (UL):

$$UL_{SDR} = 10\log_{10}\frac{\sum_n^N ||s(n)||^2 + \epsilon}{N\epsilon}$$

which is achieved when the estimate is an exact replica of the ground truth. Although being the most complete among the three metrics described in this section, the SDR still has some drawbacks. In particular, its value can be easily manipulated by means of a rescaling of the estimate. The SDR can always be positive if we scale the estimate to be the orthogonal projection of the target onto the line spanned by the estimate [64]. This is the reason why two estimates of the same source outputted by two different models can sound very different and have the same SDR. Although being the reference metric for the MDX challenge [30] and for almost every paper in the MSS research field, some improvements on the SDR have been proposed in [64] to overcome its limitations. Of particular interest is the suggestion of a new metric called Scale-Invariant Signal-to-Noise Ratio (SI-SDR). The idea is to make the SDR orthogonal to the target by either linearly rescaling the target or the source in order to get a right triangle where the

the source estimate is the hypothenuse and the cathetes are the target source and the difference between the target source and the estimate. To give a better visual representation of the difference we ask the reader to take a look at figure 3.2 which is taken from the original paper [64] and offers a two dimensional representation of this reasoning. An explicit formula for the SI-SDR is:

$$SI - SDR = \frac{|\alpha s|^2}{|\alpha s - \hat{s}|^2}$$

with $\alpha = argmin|\alpha s - \hat{s}|^2$. The rescaling parameter α can be explicitely computed as

$$\alpha = \frac{\hat{s}^T s}{||s||^2}$$

Thus leading to the following closed formula for the SI-SDR:

$$SI - SDR = \frac{||\frac{\hat{s}^T s}{||s||^2} s||^2}{||\frac{\hat{s}^T s}{||s||^2} s - \hat{s}||^2}$$

This metric offers a fairer comparison between algorithms by canceling the impact of estimate rescaling and prioritizing the direction of the estimate.

3.4.5 Subjective evaluation

Subjective evaluation of audio source separation methods involves employing listening tests to assess the quality of the separated audio. In these tests, participants listen to both the original audio mixture and the output of the separation algorithm. They're asked to rate the quality of the separated sources based on various criteria such as perceptual quality, intelligibility, artifacts, and similarity to the original sources. These assessments can be conducted through various methodologies like Mean Opinion Scores (MOS), where listeners rate the quality on a scale, or pairwise comparisons where listeners choose which audio sounds better between two versions. This last method is the one applied in the subjective evaluation part of the MDX challenge [30], where a set composed of singers, composers, music producers and sound engineers evaluated the output of the 3 models that achieved the highest average SDR across the 4 sources. Having a set of trained unbiased listeners evaluating the output of your algorithm would probably be the optimal evaluation method but it is not feasible for the majority of researchers due to its cost.

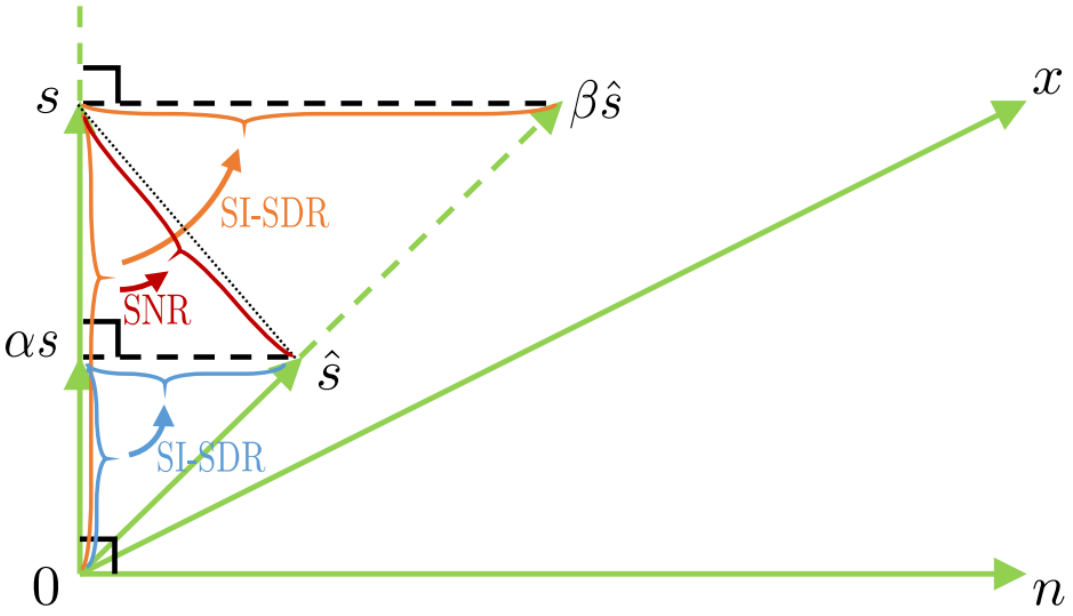

Figure 3.2: Two dimensional representation of SDR and SI-SDR. In the picture, the SDR is obtained by rescaling the estimate in order to minimize the SDR. On the other hand, in the SI-SDR framework we can see that the same rescaling implies a rescaling of the target source too, thus maintaining the same proportions. This metric offers a framework where the evaluation is solely based on the direction of the estimate, as it should be. Image from [64]

Chapter 4

Experiments

This section presents all the details regarding the experiments carried out during this Master's thesis. The first part consists of a deep dive into the training datasets, with information regarding their content and some summary statistics for relevant characteristics such as pitch, genre and others. The following sections focus on data preprocessing, experimental settings and limit estimation for this particular task. Finally, the last section is divided in several subsections, one for each of the experiments. Each subsection contains a description of the architecture used together with the reasoning behind the architectural choices and it also presents the results of the experiments.

4.1 Datasets

In this thesis, three different datasets have been exploited for the experiments. These are MUSDB18, MedleyDB-2.0 and a private dataset which has been automatically labelled using SOTA models. With the term "automatically labelled" we refer to the process of utilizing a pretrained MSS model to generate the 4 target stems from an unlabelled song. The next paragraphs contain a description of the datasets.

4.1.1 MUSDB18

MUSDB18 [65] is the most commonly used dataset for training Music Source Separation models [52][66][67] and it is composed of 150 songs with length varying from a minimum of 13 seconds up to a maximum of 10 minutes and 28 seconds, with a total length of 9 hours, 49 minutes and 19 seconds of music. The most frequent genre in the dataset is pop rock, while the most underrepresented is jazz. This imbalance should be carefully taken into consideration when analysing the performance of the model, since this highly skewed distribution could potentially

impact the generalization capabilities of the model. The compressed version of the dataset is composed 5 stereophonic stems compressed and encoded in AAC @256kbps format with a Nyquist frequency of 8kHz. This means that the samples have been collected at a sample rate of 16kHz with bit depth equal to 16. There exists a higher quality version of MUSDB18 called MUSDB18-HQ which contains uncompressed .wav files sampled at 44.1kHz. This version has not been used in order to ease the computational effort needed to train the models. Of the 5 different stems, the first one represents the mixture signal and the remaining ones represent the tracks *vocals*, *bass*, *drums* and *other*. Although some stems have been wrongfully mixed together or present some degree of source bleeding [65], these problems are limited to only 6 of the 150 songs present in the dataset. The dataset's usual split is the following: 86 songs for training, 14 for validation and 50 for testing. Due to the nature of the data, the sets are predefined in order to avoid imbalance in the validation and test songs. This should improve generalization capabilities of the trained models Figure 4.1 presents the summary graphs concerning pitch and duration of the songs in MUSDB18. The pitch plot highlights a distribution skewed towards the G pitch, either major or minor. Although this imbalance is not optimal for generalization reasons, since it is not massive we assume that its impact will not be major. The pitch has been computed by calculating the chromagram of the songs and collapsing the time dimension to obtain a 12 dimensional representation where each pitch is associated with an intensity level. The pitch with the highest intensity is selected as the pitch estimate for the song

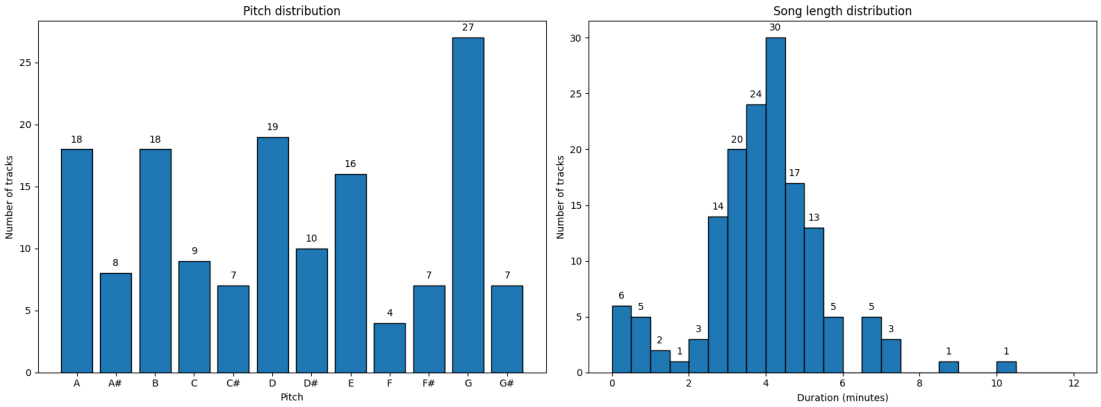

Figure 4.1: On the left: pitch distribution of the tracks in MUSDB18. On the right: length distribution of the tracks in MUSDB18

4.1.2 MedleyDB-2.0

MedleyDB-2.0 [68] is an expansion of the first MedleyDB dataset [69], part of which is included in MUSDB18. The more recent MedleyDB-2.0 is composed of 132 additional songs sampled at 44.1kHz that were not present in the previous version. Although this dataset is similar to MUSDB18 in terms of quantity of data, their contents present several characteristics that make them complementary. Indeed, in medleyDB-2.0 the most represented genre is classical music, followed by jazz, folk and songwriting [68], a distribution skewed towards the underrepresented genres in MUSDB18. While this makes the datasets complementary for Genre Classification tasks, the most part of MedleyDB-2.0 is not suitable for the standard MSS goal of separating the stems *vocals*, *bass*, *drums* and *other*. This is due to the fact that in most songs one of the 4 stems is missing. The number of stems in each song varies from 3 to 10 and the metadata related to the content of the stems can be found in the gitHub page of the dataset [70]. By exploiting the metadata we were able to reduce the number of stems for each song to a maximum of 4 stems. The *vocals* stem has been generated by adding together all the stems named either *choir, female singer, male rapper, male singer* or *vocalists*. The *bass* one was generated by changing the name of the *electrical bass* stem in MedlyDB-2.0. The *drums* track is the result of the sum of the *bass drum, bongo, drum machine, drum set, high hat, kick drum* and *snare drum* stems. All the remaining ones have been grouped together and summed to generate the *other* stem.

4.1.3 Private Dataset

The first version of this private dataset is composed starting from 160 songs in .mp3 format at various bit rates. These songs have been selected with a limit of 5 songs per artist in order to avoid overfitting on a single voice or musical style. Although these songs do not come with genre metadata, the goal when composing the dataset was to keep it as heterogeneous as possible. This meant trying to balance the addition of rock, pop and blues songs while also taking into consideration whether the singer was a male or a female. This can have a profound impact on vocals separation since male singers usually have a lower frequency range that overlaps with instruments, while female singers usually sing in the higher frequency range. Due to this intrinsic characteristic of the stem, both groups require adequate representation in the dataset in order to train a model that is able to generalize well. The stem targets for these .mp3 mixture files have been created using the output of the state of the art model Hybrid Transformer Demucs [59]. Although these targets are just an estimate of the optimal ones, the goal of the dataset expansion is to properly evaluate the implementation of a self-attention based model, which is notoriously a data-hungry mechanism [71] and thus requires more training samples to reach its potential. In the second version of the dataset, 100 additional songs

have been added following the same procedure. The only difference is the use of the fine-tuned version of HT Demucs for the target stem generation. This should result in higher quality of the training data since the fine-tuned version achieves a 0.2dB improvement on the median SDR metric [59].

4.2 Data Processing

The first step of the data processing phase is the downsampling of all the training, validation and test songs to a 16kHz sampling rate using the resampling function of the librosa library [72]. The choice of librosa's resampling function comes after an auditory test of the performance of both librosa's and torchaudio's resamplings. The latter resulted in a less balanced result for the mixture, with some of the background instruments almost disappearing from the audio file. Keeping the same bit depth of 16 bits, the resampling resulted in a 2.76x reduction of the memory required to store the data which then translated into a faster training procedure.

After the training data have been downsampled, the subsequent data processing steps vary between the training set and the validation and test sets. Following instructions from [52] I performed one of the most recent Source Activity Detection (SAD) procedures on the training data. Each song has been divided into 6 seconds long audio chunks with a 50% overlap. Each chunk has been subsequently divided into 10 segments of length 0.6 seconds with no overlap, which correspond to 9600 samples for an audio file sampled at 16kHz. After that, the RMSE is computed for every segment following equation 2.5. Segments with RMSE values equal to zero (i.e. completely silent segments) have their energy set to the arbitrary low number of 1^{-5}. The threshold for deciding whether a chunk is silent or not is defined as the value of the 15th quantile of the energy of all segments in the processed song. An audio chunk is then defined as salient if 50% of its 10 segments have an energy higher than the computed threshold. While this could seem like a tedious task and a waste of training data, the effects of SAD will be evaluated as part of an ablation study during the first experiment. The SAD has been performed on both the MUSDB18 dataset and the MUSDB18 dataset with the added 260 songs from the private dataset. Results are presented in table 4.1 where we can see that the SAD marked as salient an average of 83.52% of the audio chunks in MUSDB18, while for the extended dataset this percentage is 86.62%. It should be noted that although the stem *other* has the highest percentage of salient chunks, this is not necessarily an advantage because of the intrinsic diversity of instruments that populates this particular stem. If the instruments for this stem are not consistent throughout the dataset, having more salient chunks could actually turn out to be a disadvantage for the models.

The validation and test sets have been divided in non overlapping 6 seconds audio

	Stem				
Dataset	vocals	bass	drums	other	overall
MUSDB18	68.44%	83.84%	86.17%	95.64%	83.52%
Private v2	79.61%	82.92%	86.30%	97.64%	86.62%

Table 4.1: Percentage of audio chunks labelled as salient during the SAD phase. The results are presented for the MUSDB18 dataset and for the MUSDB18 dataset expanded using the entirety of the private dataset, which we refer to as Private v2.

chunks and no SAD has been applied in order to give a fair representation of the generalization capabilities of the model.

4.3 Experimental settings

The tests have been carried out on two different hardware settings in order to speed up the procedures. The experiments involving RMSE, ZCR or the use of self supervised features have have been carried out on a nVidia Tesla V100 SXM2 with 32 GB of memory. These computational resources have been kindly provided by hpc@polito, which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino [73]. All the experiments that use of the magnitude spectrogram as the only input, have been carried out on a nVidia GeForce GTX 1650 GPU with 4GB of memory. Trainings have been performed for 20 epochs with a batch size of 8 and a simple stopping criteria based on the validation loss: if the validation loss increases with respect to the previous epoch, training is stopped. The loss function varies from one experiment to the other and will therefore be made explicit in the appropriate sections. The optimizer chosen for all the experiments is the AdamW optimizer with weight decay equal to 0.0001, $\epsilon = 1^{-6}$, $\beta_1 = 0.9$ and $\beta_2 = 0.99$. Following the instructions in [74], the learning rate starts from 0 and is updated following a learning rate scheduler defined as follows:

$$LR = d_{model}^{-0.5} \cdot min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

where $warmup\_steps$ is set to 4000 and d_model is the dimension of the encoder's input features. The scheduler step happens at every iteration. When using a batch size of 8 and MUSDB18 as the training dataset, the linear warmup phase lasts until the 8th epoch. When using the expanded version of the dataset, the warmup ends in the middle of the 2nd epoch. Although this setup covers the majority of the experiments, some slight modifications have been applied to further improve performance. In this case, all the details regarding the modified configuration will be noted in the corresponding experiment's section. For validation and testing

we used the validation and test sets of MUSDB18. The results are presented in the form of a metric that we will refer to as the median SDR in order to ease the notation. To compute this metric we first compute the stem estimate for each song in the test set, we then split this estimate into non overlapping 1 second segments and we compute the SDR for each of them. We take the median of these SDRs, thus obtaining a median SDR for each song in the test dataset. Finally, to compute the median SDR metric we take the median of these 50 values. The choice of this metric follows instructions reported in SiSEC18, i.e. the Signal Separation Evaluation Campaign of 2018[75], and makes comparison with other SOTA algorithms easier and more meaningful. The training times for each configuration tested in this section are presented in appendix C.

4.4 Model's Limits Estimation

To contextualize the model's performances, we defined an Upper Limit (UL) and a Lower Limit (LL) for the median SDR metric in this experimental setting. To define the LL we generated a random mask for every non overlapping 6 seconds audio chunk in the test set and we multiplied it by the mixture's magnitude spectrogram, resulting in a random prediction of the target stem's magnitude spectrogram. Using this prediction and the phase of the mixture, we then generated a complex spectrogram and performed an iSTFT on it to obtain a wave prediction. We then computed the median SDR metric following instructions in section 4.3 and set the it as the LL.
For the UL estimation, we generated the complex spectrogram using the target stem's magnitude spectrogram and the mixture's phase. Using the target stem's magnitude spectrogram will result in the best prediction that our model could possibly achieve. The rest of the procedure for the UL estimation is the same as the one for the LL. We repeated the UL and LL computations for each of the four stems. The performance limits are presented in table 4.2, where we can notice that the *bass* source is expected to be the one with the lowest values of the median SDR, followed by *other*, *drums* and *vocals*. This is in line with the literature, where the median SDR on *vocals* is often the highest between the 4 stems[52][51][76].

4.5 Frame-Wise Attention Mechanism

4.5.1 Baseline Experiment

For the first experiment, to the 6 seconds audio chunks of the mixture and of the target we applied a STFT with parameter nFFT equal to 2048 and hop length equal to 1024. The window used is a Hann Window of size 2048 obtained by

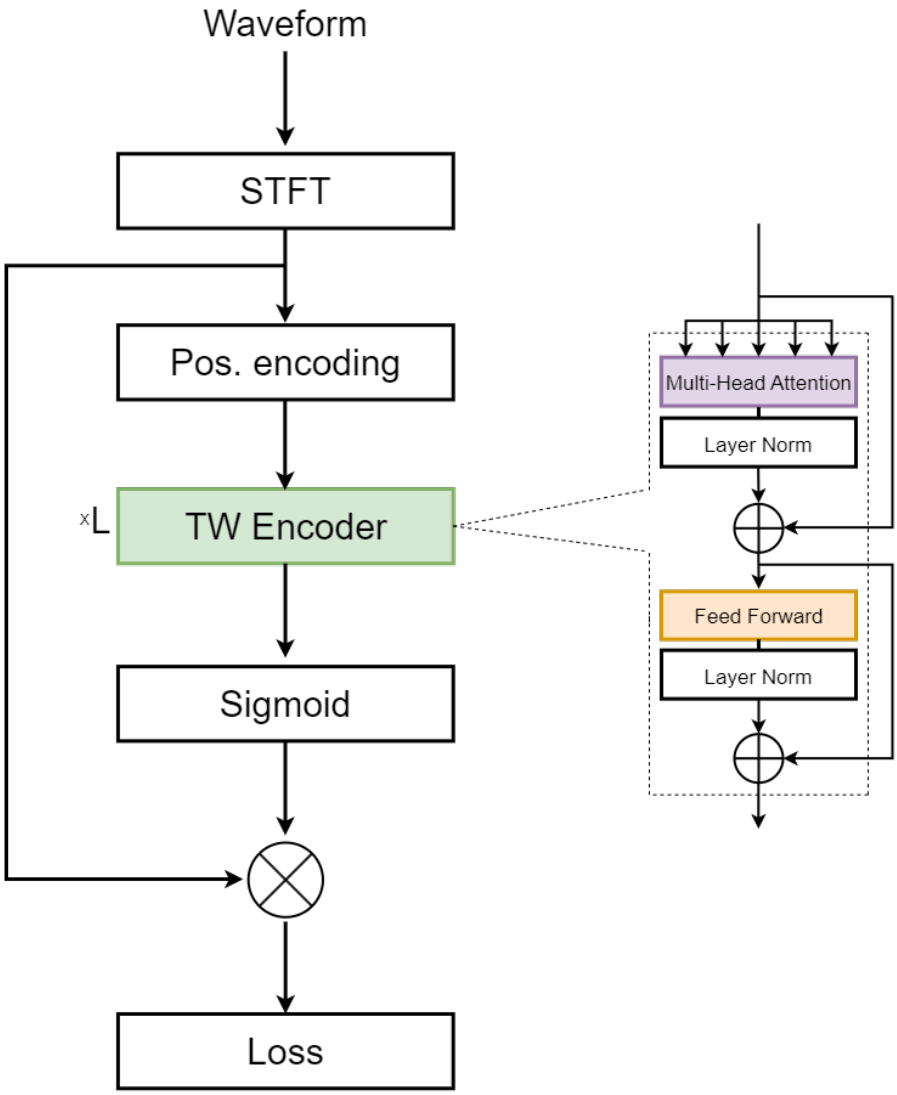

Figure 4.2: Architecture of the base experiment. The Time-Wise Encoder has been implemented following the the description found in [74]

	Limit estimate	
Stem	LL	UL
Vocals	-6.620	9.843
Bass	-6.344	5.981
Drums	-3.979	8.751
Other	-5.273	6.859

Table 4.2: Lower Limit (LL) and Upper Limit (UL) estimation for the model's performances

creating a Hann Window of size 2058 using the corresponding torch function and discarding the first and last 5 elements. This additional step has been done because with a typical Hann Window formulation, torch's inverse STFT function which is used during testing would result in a RunTime Error triggered by the fact the the boundary values are too low. Due to the nature of the Hann Window function, the number of samples to discard at the edges of the window is lower with lower values of the nFFT parameter. For example, with nFFT=400 the number of edge samples to discard to avoid the error is 1. The STFT has been performed with centered time frames and constant padding with padding value equal to 0. The result of the STFT operation is a complex spectrogram with 1025 frequency bins which spans across 94 frames. Given that the sample rate of the audio files is 16kHz, the Nyquist frequency is 8kHz and therefore each spectrogram bin describes a frequency range of 7.8kHz. While this thesis tests a new approach for phase estimation in the last section, in this experiment target phase estimation is not among the goals and the focus is instead on magnitude spectrogram estimation. For this reason, to the complex spectrogram we apply the absolute value and extract the magnitude spectrogram of the mixture, which will be the input of the model. The phase information needed to perform iSTFT during testing will come from the mixture's phase which, although suboptimal, is nevertheless an acceptable estimate of the target's phase and allows evaluation of the model's performance in real case scenarios where optimal phase information is not available.

The proposed model's architecture is based on the Transformer Encoder [74] with some minor adjustments, as figure 4.2 shows. The input is of shape *(B, T, F)*, where B is the batch size, T is the number of time frames and F is the number of frequency bins. In this first experimental setting B=8, T=94 and F=1025. The embedding phase typically used in Transformer architectures is skipped since it is not suitable for audio data. Therefore, the raw mixture's magnitude spectrogram is positionally encoded frame-wise using sine and cosine functions of different frequencies [74]:

$$PE(pos,2i) = sin(pos/10000^{2i/F})$$

$$PE(pos,2i+1) = cos(pos/10000^{2i/F})$$

where i is the time frame index ranging from 1 to T. The encoded spectrogram is then passed through L Time-Wise (TW) Transformer Encoder Layers, where L is the number of layers and is initially set to 6. The Encoder layer is composed as the one in the original paper: first, multi-head self-attention with 5 heads is applied to the input, the resulting projection is then normalized and added to the input using a skipped connection. This intermediate representation passes through a feed forward with a single hidden layer of dimension 2048 that uses ReLU as the activation function. Layer normalization is then applied to the result, which is added to the previously computed intermediate representation using a second skipped connection. The output of the encoder layer will function as the input of the next layer. After passing through the L Encoder layers, a sigmoid function is applied to the output of the last layer in order to bring its range between 0 and 1 and allow us to use the result as a multiplicative mask for the input magnitude spectrogram.
This architecture aims at evaluating the impact of a Transformer Encoder (and, in particular, its Multi-Head Attention block) in Music Source Separation, with the goal of understanding whether its capabilities in modelling sequential data can help overcome the computationally expensive U-Net architectures which are at the base of many SOTA models [66][59][77]. The model is trained 4 times, once for each target source, using Mean Absolute Error (MAE) loss defined on the target stem's magnitude spectrogram. Defining as $\hat{S}$ the spectrogram estimate obtained with our model, the loss is defined as

$$Loss = \frac{1}{N}|\hat{S} - S|$$

where N is $B \times T \times F$. Following the Transformer paper's guidelines, the first experiment was performed using L=6 and is used as the baseline. The results are presented in table 4.3 in the form of median SDR for each stem. While the model seems to obtain below average results on *vocals* and *drums*, it struggles even more in the estimation of the *bass* and *other* stems. This last point is in line with the literature, where the *bass* and *other* often appear to be the most difficult stems to predict [78][52].
To understand the cause of the low performance compared to SOTA models, we delved into an analysis of the SDR distribution to find out which audio characteristics correlate with lower performances of the model. As figure 4.3 shows, the SDR distribution of *vocals* predictions shows two peaks: one centered at around -25dB

	Stem			
Architecture	Vocals	Bass	Drums	Other
Baseline (L=6)	2.881	-1.903	2.454	-0.919
Layers = 3	2.453	1.384	2.826	-1.542
Layers = 2	**4.257**	**1.691**	**3.835**	0.366
Layers = 1	3.564	1.260	2.971	**0.687**

Table 4.3: Median SDR across the test set for every stem with different numbers of encoding layers

and one centered at around 5dB. This suggests that some characteristic or set of characteristics of the mixture data might split the test set in two groups, one of which is more difficult for the model to separate.

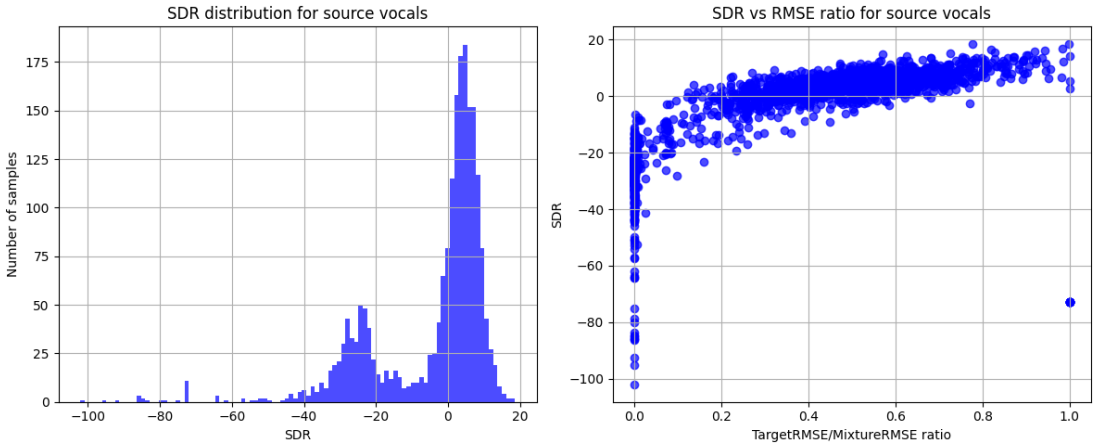

Figure 4.3: On the left: SDR distribution across the test set. On the right: SDR plotted against the ratio between the target wave SDR and the mixtue wave SDR.

By performing an auditory test on the audio chunks that generate SDR values lower than 0, we found out that most of these audio chunks are either silent or partially silent for the target stem. To better investigate this correlation, for each chunk in the test set we plotted the SDR values against the ratio between the target and mixture waves' RMSEs. Low values of this ratio indicate an audio chunk where the model needs to predict a silent or partially silent stem starting from a non silent mixture. When the values of this ratio are close to 1, the corresponding audio chunk contains a mixture and a target stem that tend to overlap, thus making the separation task trivial. The results of this analysis are presented in figure 4.3, where we can appreciate a positive correlation between the performance of the model and the RMSE ratio we just described. This confirms the hypothesis that

the model struggles to predict silent chunks. It must also be noted that the outlier seen in the plot (ratio value close 1, SDR value close to -75dB) represents a silent audio chunk, i.e. both the mixture and target are silent. The auditory analysis of SDR values in the higher end of the SDR distribution also showed that high SDR values correspond to audio chunks that are more likely to feature a female singing voice. This probably originates from the fact that female voices have a higher vocal range and are thus easier to separate from the rest of the mixture since there is less overlap in the frequency of the different stems. Other problems noticed during the auditory test are the muffled sounds for the both the bass and drums predictions and the source bleeding which is a strong component of this model's predictions. As a final remark, we would like to point out that while the plots in figure 4.3 are related to the *vocals* stem, similar graphs have been obtained for *bass*, *drums* and *other* but have not been presented to avoid redundancy in the report. The reader can find these plots in appendix D.

4.5.2 Depth Reduction

Knowing that the self attention mechanism from the transformer architecture is considered "data-hungry" [71], the first attempt at improving the model's performance focused on tweaking the ratio between the number of training data and the number of trainable parameters in the model, which amounted to 58.9M for the baseline. To increase this proportion we tried lowering the number of Encoder layers.
We tested 4 different configurations: the first one is the baseline with L=6 the other three are the ones with L=3, L=2 and L=1. As table 4.3 shows, the model does indeed benefit from a reduction in the number of attention layers, since the highest performance across all configurations was obtained with L=2, i.e. 25.2M parameters. The source that benefits more from the reduction is *bass*, with an increase of 3.59dB in the median SDR of its predictions. An auditory test confirms the improvements and highlights a lower amount of muffled sounds for the drums and bass sources. It should be noted that a model with L=1, although preferable to a L=6 configuration, is apparently too simple to model this kind of data since the performance does not seem to improve as much.

4.5.3 Dataset Expansion

The second attempt at improving the training samples to parameters ratio focused on the expansion of the training set. To achieve that, the first 160 songs of the private dataset have been used as additional training data to the model, while keeping the same validation set. The results are presented in table 4.4, where we can appreciate an improvement of 0.16dB for *vocals*, 0.54dB for *bass*, 0.61dB for *drums*

and 0.95dB for *others*. Encouraged by the results, we further expanded the training set by adding 100 additional training songs separated using Hybrid Transformer Demucs [59], which should yield slightly greater quality of the automatically created target stems. As table 4.4 shows, further increasing the training set resulted in a leap in performance, with reported increments compared to the baseline of 0.66dB for *vocals*, 1.60dB for *bass*, 0.83dB for *drums* and 0.62dB for *other*. No further dataset expansion has been performed for the training set since we decided that the collected results were enough to evaluate the impact of data quantity for this field of research. All experiments from now on will use the expanded dataset for training in order to properly evaluate the proposed modifications.

	Stem			
Training DB	Vocals	Bass	Drums	Other
MUSDB18	4.257	1.691	3.835	0.366
Private v1	4.415	2.237	4.441	**1.318**
Private v2	**4.914**	**3.299**	**4.660**	0.984

Table 4.4: Median SDR response to the training set expansion. Private v1 refers to the dataset composed of MUSDB18 and 160 additional songs from the private dataset. Private v2 is composed by Private v1 and 100 more songs.

4.5.4 SAD Impact Evaluation

The last two attempts at improving the model's predictive abilities are motivated by the analysis of the SDR distribution presented in figure 4.3. The model seems to struggle in the prediction of silent or partially silent audio chunks and one of the possible root causes is the SAD applied to the training data, which causes a lack of silent training samples and may therefore be the reason behind the model's struggles. To verify this assumption we repeated the experiments with batch size equal to 8 without applying SAD to the training set. The results are presented in table 4.5 where we can appreciate a drop in performance, confirming that training data for Music Source Separation need filtering in order to avoid confusing the model with lower quality samples.

4.5.5 SISDR Base Loss Function

The last attempt at improving the model's abilities in silent chunks separation is a modification of the loss function. Compared to the original MAE loss, the new loss features an additional SISDR component, which is applied to the iSTFT of the complex spectrogram generated using the predicted magnitude spectrogram and the mixture's phase. The new loss is therefore defined as follows:

	Stem			
Architecture	Vocals	Bass	Drums	Other
With SAD	**4.914**	**3.299**	**4.660**	**0.984**
Without SAD	4.787	2.990	3.495	0.846

Table 4.5: Median SDR one the test set for the model trained on processed vs. unprocessed training data

$$Loss = MAE(\hat{S} - S) - SISDR(iSTFT(\hat{S}, P), W) \tag{4.1}$$

Where W is the target wave and P is the phase of the mixture. The minus sign has been added because good predictions are characterized by high values of the SISDR and the model has been trained to minimize the loss function. Since the SISDR is differentiable, it can be safely used in the loss function and it should help the model to focus more on silent chunks, which are characterized by very low values of the SISDR. The results are presented in table 4.6 and we can see that the new loss function actually causes a drop in performance compared to the baseline. The SDR distribution presented in figure 4.4 also highlights the missed shift in the SDR distribution, a distribution that we expected to converge to a more normal-like one. It still shows two distinct peaks in similar positions, meaning that the loss change did not serve its intended purpose. Although the overall performance suffered from the implementation of the modified loss function, we should also notice that the SDR distribution seems to have less extreme negative values compared to the one in figure 4.3

	Stem			
Loss function	Vocals	Bass	Drums	Other
Standard Loss	**4.914**	**3.299**	**4.660**	**0.984**
NewLoss	4.017	2.741	3.614	-1.665

Table 4.6: Median SDR on the test set for the model trained with standard loss and with the modified loss featuring a SI-SDR component

4.5.6 True Performance Estimation

Using the expanded dataset, in order to find the true performance ceiling of the model we repeated the training with a much higher batch size of 128 and an expanded validation set. The validation set has been expanded using the following 8 tracks collected from MedleyDB-2.0: 'Hunting Season' by Midnight Blue, 'Perfect Day' by Cassandra Jenkins, 'Prisoners Cinema' by Dead Milkmen, 'Stars Are

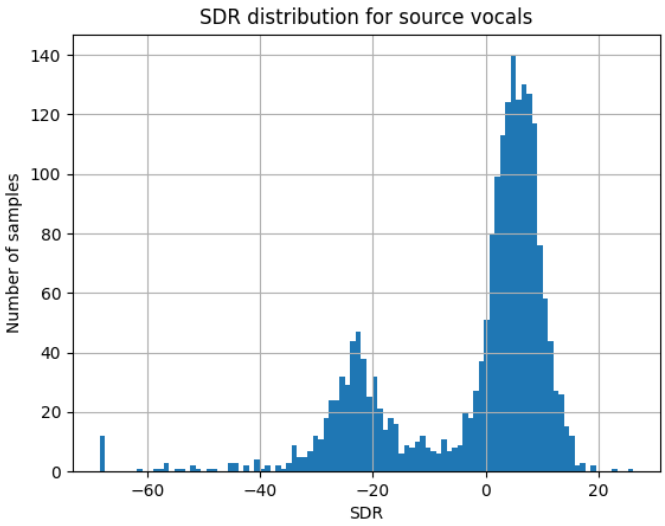

Figure 4.4: SDR distribution for the test set audio chunks. Predictions generated with the modified training loss described in equation 4.1

Screaming' by Midnight Blue, 'Lush' by The TonTons, 'New Skin' by Torres, 'Alone And Sad' by Trevor And The Soundwaves and 'I'd Like To Know' by Filthy Bird. These songs have been selected following two simple criteria: the songs must possess all 4 target stems and no more than two songs per artist can be selected. The aim of this expansion is to improve the generalization capabilities of the model with an indirect modification of the stopping criteria, which will now be computed on a more representative and diverse validation set.

We can observe the impact of increased batch size and extended validation set in table 4.7. As we can see, the baseline performance for *bass* and *drums* appears to be already close to optimal since the two tested factors either resulted in a marginal gain or in a loss in performance. On the other hand, *vocals* saw a 0.39dB improvement in the median SDR metric when using the expanded validation set. The biggest improvements come in *other* separation, where both factors seem to be highly useful for the estimation and even more so when applied simultaneously . Although some researches work with larger batch sizes for more training epochs[59], a batch size of 128 should be big enough to leave very small margins of improvements over the current performance. Moreover, a SOTA model such as Band-Split RNN[52] achieved great results with much smaller batch size, meaning that architectural choices have far greater impact on performance with respect to the batch size.

	Stem			
Architecture	Vocals	Bass	Drums	Other
Baseline	4.914	3.299	**4.660**	0.984
BatchSize=128	5.076	**3.372**	4.553	2.073
Expanded Val	**5.306**	3.192	4.335	1.620
Both	5.152	3.266	4.404	**3.058**

Table 4.7: Median SDR obtained with a greater batch size and an expanded validation datase

4.6 Frame-Wise and Frequency-Wise Attention Mechanisms

The architectures for this experiment are presented in figure 4.5.

4.6.1 Parallel TW-FW attention architecture

Inspired by the work performed in [52], we decided to implement both a Time-Wise (TW) and Frequency-Wise (FW) Multi-Head Attention mechanism to evaluate the impact of the FW component on the separation abilities of the model. The first model implementation features an architecture with two branches: a TW branch and a FW branch. The first one is an exact replica of the architecture in figure 4.2 and outputs a mask that we will refer to as the TW mask. The second branch features a permutation of the spectrogram and skips the positional encoding of the frequency vectors because they are not sequential elements. As for the TW branch, the FW one features 2 Encoder layers with a Multi-Head Attention mechanism with 2 heads and processes a batch of shape (B, F, T), where F is 1025 and T is equal to 94 which is the number of frames obtained with a centered STFT applied with nFFT=2048 and hop length equal to 1024. The Multi-Head Attention is part of the Transformer Encoder unit and for this reason it is followed by layer norm, a feed forward with a hidden layer of dimension 2048 and another layer norm. The Transformer Encoder is followed by a sigmoid function to generate a mask, that will be referred to as the FW mask. The TW and FW masks are summed together with a weighted sum, where the weights are defined as the softmax of two learnable parameters that are initialized to 1 at the beginning of the training phase. The final mask is then multiplied by the input magnitude spectrogram of the mixture to obtain a prediction.
The model has been trained with batch size equal to 8 on the MUSDB18 training set with the private dataset expansion. The validation set is the standard MUSDB18 one. For these reasons, the most suitable baseline results for this architecture are

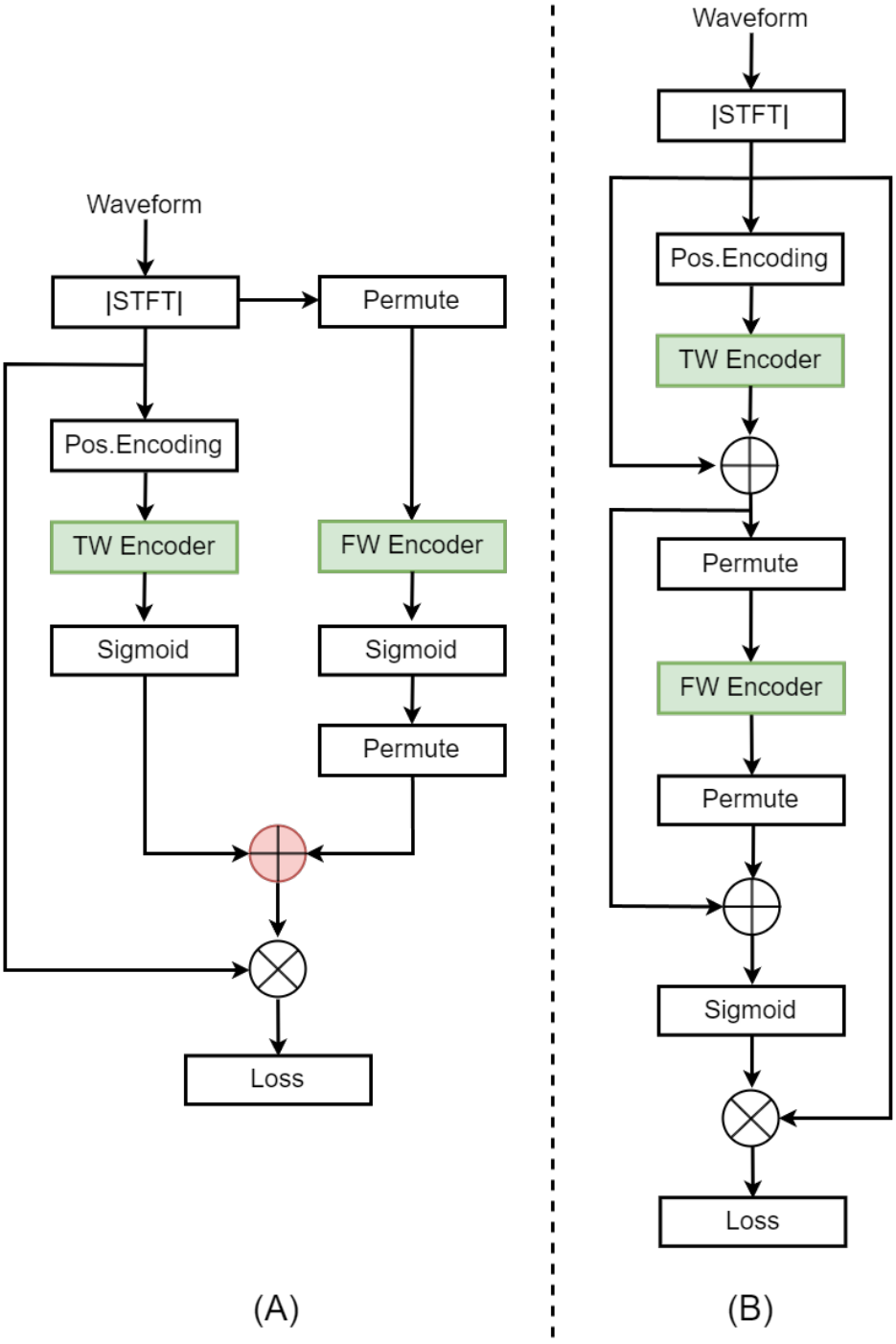

Figure 4.5: *On the left*: parallel TW-FW attention architecture. *On the right*: consecutive TW-FW attention architecture

the one presented in the last row of table 4.4.
The performance of the model is summarized in table 4.8. The addition of the FW encoder resulted in a loss of performance in 3 out of 4 stems, with *other* being the only one that benefited from the architectural modifications. The degradation in performance may be due to confusion generated in the model by the second branch, since the model should be able to at least replicate the baseline results by setting the weighted sum's weights for the TW and FW masks to 1 and 0, respectively.

	Stem			
Architecture	Vocals	Bass	Drums	Other
Baseline	**4.914**	**3.299**	**4.660**	0.984
Parallel TW-FW	3.924	1.712	3.497	1.728
Consec. TW-FW	4.085	2.226	3.645	**2.171**

Table 4.8: Median SDR on the test set for the parallel TW-FW attention and consecutive TW-FW attention architectures. The Baseline model is the one from table 4.4

4.6.2 Consecutive TW-FW attention architecture

The second architecture mimics more closely the one in [52], substituting the BLSTM components with Transformer Encoders. The result is a model structure similar to the one referred to as Time-Frequency Conformer in [61], with the substitution of the conformer block with a Transformer Encoder one. As shown in figure 4.5, the model's pipeline starts with the positionally encoded magnitude spectrogram of the mixture which is passed through a TW Transformer Encoder. The resulting intermediate representation is summed to the input magnitude spectrogram using a skipped layer, yielding a second intermediate representation. This is subsequently rearranged for input into an FW Transformer Encoder, and then permuted back to align with the original spectrogram shape. The outcome of this Encoder is summed with the second intermediate representation using a skipped layer. The result is then fed through a sigmoid function to generate a mask, which, when multiplied with the magnitude spectrogram of the mixture, produces an estimation of the target stem. The TW-Encoder has been set to have 2 layers with a number of attention heads equal to 5, while the FW-Encoder has 2 layers and 2 attention heads. Both use a feed-forward with hidden dimension equal to 2048 and apply layer norm after both the multi-headed attention the feed forward. Results are presented in table 4.8 and are similar to the ones obtained with the first architecture. The performance is far worse than the one obtained with the baseline but we can notice a substantial increase in *other* and *bass* performance of

0.44dB and 0.51dB, respectively. Predictions for the other stems show marginal improvements, with a 0.16dB increase for *vocals* and a 0.15dB increase for *drums*.

4.7 Self Supervised features

4.7.1 WavLM Base+

In this experiment we will investigate a method for integrating features generated by the Self-Supervised general pre-trained model WavLM Base+ [28]. The architecture for this experiment is represented in figure 4.6.

	Stem				
Architecture	Vocals	Bass	Drums	Other	Average
Baseline	2.072	0.078	1.807	0.149	1.026
WavLM last layer	2.068	-0.164	2.588	0.087	1.145
WavLM weighted sum	2.247	1.118	3.769	0.375	1.877
Hubert Base Audioset	2.477	1.084	**4.028**	0.463	2.013
Hubert Large Audioset	2.330	**1.550**	3.242	0.930	2.013
Wav2Vec2 Base Audioset	**2.607**	1.492	3.046	**1.030**	**2.044**
Wav2Vec2 Large Audioset	1.882	0.747	3.003	0.401	1.508

Table 4.9: Median SDR on the test set for the models using the architecture in figure 4.6. A column named average has been added to make comparison between models easier. It contains an average of the 4 medians presented in the remaining columns.

The mixture waveform is processed in two different branches to extract different types of information from it. The first branch is the one seen in the previous experiments, where the waveform is passed through a STFT with hop length equal to 160 and number of FFTs equal to 400 to match the output of the SSL model. An absolute value is applied to the complex spectrogram to obtain the corresponding magnitude spectrogram. In the second branch, the waveform is processed using the WavLM Base+ model, which receives an input of shape (B, SR*6) and generates an output of shape (B, T/2, 768), where B represents the batch size, SR the sample rate of the wave and T represents the number of time frames generated in the STFT. To make the number of features match we could have selected a hop length of 320 for the STFT but we chose not to because of he very low number of FFTs, which could hinder the performance of a MSS model. We therefore chose to avoid doubling the hop length to 320 to prevent further deterioration of the features collected in the spectrogram branch. In order to make the number of features match we decided to follow the strategy described in [79] and halve the stride of the

last convolution layer, thus doubling the number of features generated. The two features obtained from the two branches are concatenated together into a tensor of shape (B, T, 201 + 768). This is then fed into a linear layer which brings the data into a mask-like shape of (B, T, 201). The usual positional encoding is applied to the data, which is then passed through a TW encoder. The rest of the algorithm is the same, with a sigmoid function applied to generate the mask, which is then multiplied to the input magnitude spectrogram to obtain the prediction for the target stem. The second version of the algorithm is the same except for the SSL branch. Instead of using the output of the last encoder layer, this version utilizes the outputs of all the 13 layers (12 Encoder layers and the intial convolutional encoder) to generate a representation of the waveform. The 13 representations of shape (B,T,768) are summed using a weighted sum with weights defined as the softmax of a set of 13 learnable parameters that have been initialized to 1 at the beginning of the training phase. From the concatenation onward, the remaining operations are the same as the ones we just described. To obtain a fair evaluation of the impact of the introduction of the SSL features on the model's performances, we trained a baseline architecture with the same STFT parameters without the SSL branch. The linear layer is still present but is now a 201×201 linear layer. Results are presented in table 4.9. We can appreciate an increase in model performance for *vocals* and *bass* when using the representations from all 13 layers. On the other hand, the predictions for *drums* and *other* are significantly damaged by the SSL branch. Results for these last two stems are more similar to the baseline for the model using only the last layer of the SSL model. On the other hand, the predictions for *vocals* and *bass* reported a decrease in median SDR of 0.45dB and 0.29dB, respectively.
We also present an analysis of the weights assigned to the 13 layers. Their values at the end of the training can be observed in the heatmap in figure 4.7. As evidenced by the plot, the last layer is associated with the highest weight on all 4 stems. Starting from the last layer, the associated weight follows a decreasing pattern as we approach the fourth to last layer. Layers before that show little to no importance in the computation of the aggregated feature. The only exception is the *vocals* stem, where layers from 1 to 9 are associated with small but not negligible weights. The stem *other* is the one where the first layer is associated with the highest weight, which is equal to 0.288.

4.7.2 HuBERT and Wav2Vec2 trained on Audioset

This experiment will use the same architecture observed in figure 4.6 but instead of WavLM Base+ we will use new versions of HuBERT and Wav2Vec2 which have been trained on Audioset, a collection of several audio related datasets. We will test the Base and Large versions of these two models, thus effectively testing four

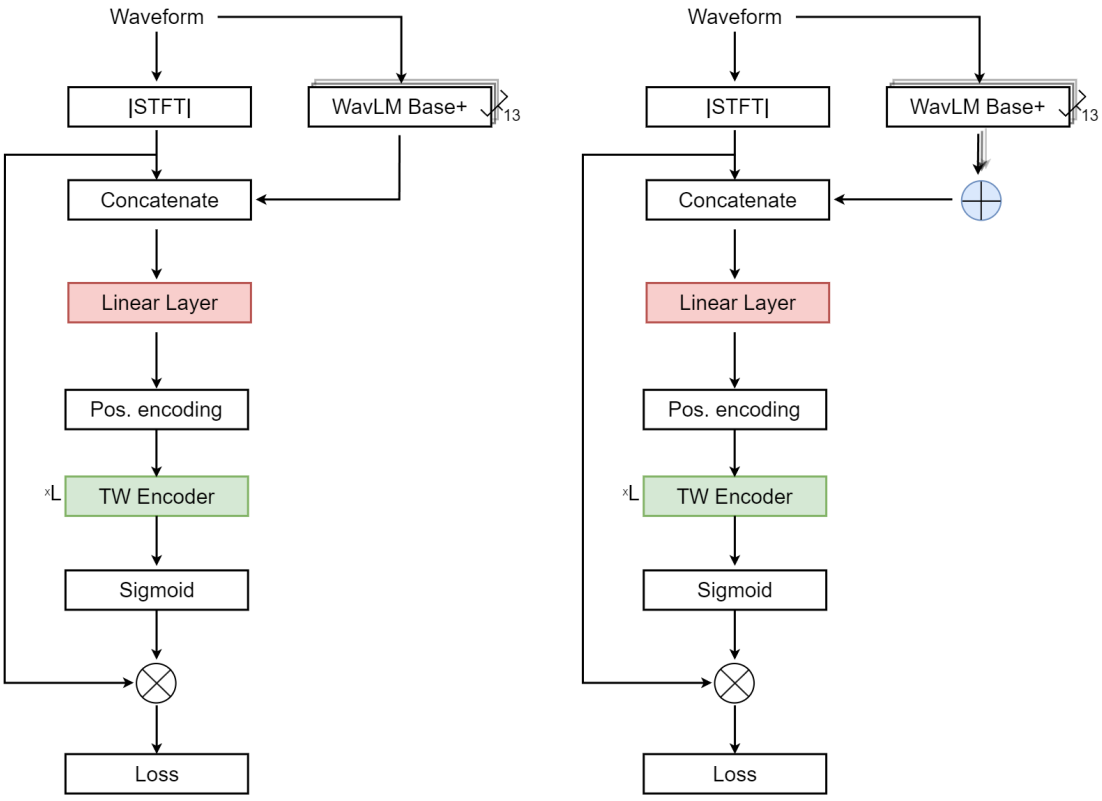

Figure 4.6: *On the left*: architecture of the model with SSL features which utilizes the last layer .On the right: architecture of the model with SSL features which utilizes the last layer

different configurations. The feature generated by the SSL branch for these four experiments will be the weighted sum of all the features generated by the models. We chose to follow this strategy since it is the one that gave the best results on WavLM Base+. The results are presented in table 4.9 to offer a direct comparison to the results obtained with WavLM Base+. We can see that both the Base models outperform WavLM Base+, with Wav2Vec2 Base yielding the best results on *vocals* and *other*. Regarding their Large counterparts, the HuBERT model performs on average as its corresponding Base model but with significantly better performance on *bass* and, on the other hand, significantly worse on *drums*. The Wav2Vec2 Large model appears to bring improvements over the baseline but struggles when compared with the other 3 models and with WavLM Base+.

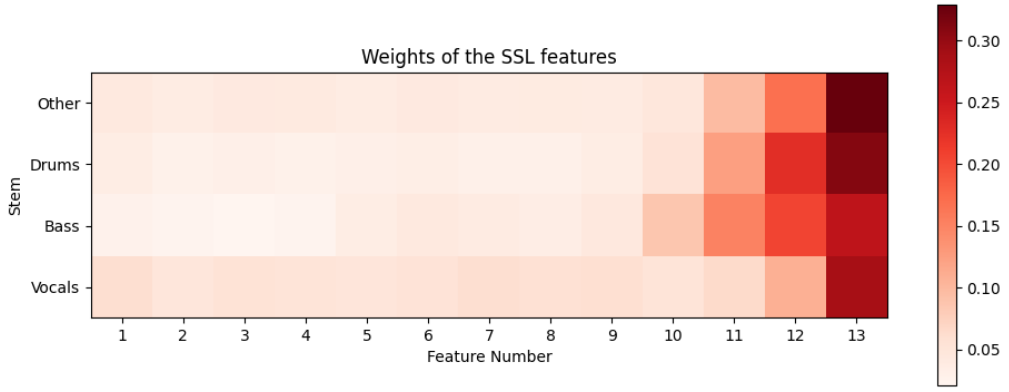

Figure 4.7: Heatmap representing the weight given to each layer in the weighted sum performed over the SSL features. The weights are presented for every stem and are computed as the softmax of the 13 trainable parameters assigned to the features

4.8 Classic audio features

This experiment is very similar to the previous one and for this reason we will refer to image 4.6 for the architecture representation. The only difference to keep in mind is that this experiment does not use SSL features, so instead of using the WavLM Base+ model, the waveform is processed using ZCR and RMSE for every frame. The branch that we referred to as the SSL branch will now be called the secondary branch and will output either one or two (B,T,1) shaped tensors representing the ZCR and RMSE of the input waveform. The concatenation with the magnitude spectrogram will bring the shape of the data to either (B,T,F+1) or (B,T,F+2) and a linear layer brings the dimension back to (B,T,F) so that it could be fed into the rest of the model, which remains the same. The first configuration tested uses ZCR, the second one uses RMSE and the last one uses both of them simultaneously. The baseline model for this experiment does not include the secondary branch but features a $F \times F$ linear layer anyway to make the comparison fairer.
We can appreciate the results in table 4.10. The introduction of RMSE seems to offer the most stable results, with an increase in performance across all the 4 stems.

The *other* is the one that benefitted the most from the concatenation of the RMSE, showing an increase of 0.31dB in its median SDR value. ZCR, on the other hand, achieves greater improvements on *vocals* and *bass* stems, which increased by 0.40dB and 0.30dB, respectively. On the other hand, the introduction of ZCR appears to give less consistent results on the 4 stems since it generated a drop in performance of 0.43dB for *drums* and 0.49dB for *other*. A similar behaviour is followed when concatenating both features to the spectrogram. In this case, performance on *vocals* and *bass* increased with respect to the baseline, while *drums* and *other* suffered from a loss in performance, although not as severe as the one observed when using only the ZCR.

	Stem			
Architecture	Vocals	Bass	Drums	Other
Baseline	4.138	2.316	4.230	1.579
ZCR	**4.535**	2.610	3.802	1.086
RMSE	4.344	2.598	**4.301**	**1.889**
Both	4.481	**2.719**	3.883	1.211

Table 4.10: Median SDR on the test set for the models using the ZCR and RMSE features.

4.9 Imaginary part management

This section of the chapter will focus on the attempts that we made to deal with the imaginary part of the complex spectrogram. In our first unsuccessfull attempt we concatenated the real and imaginary parts to obtain spectral features of shape (B, T, 2F). We then implemented a Transformer Encoder configured to work with vectors of length 2F and instead of predicting a mask we tried to predict the exact spectrogram. The model's performance fluctuated around -4dB for the median SDR of the various stems. We tried changing several parameters in the training, including experiencing with different optimizers like Lion[80] and a Sharpness Aware Minimizer[81] applied to AdamW but we found no success at all. We therefore decided to shift our focus to the mask estimation approach that we used in the previous experiments and decided to work with magnitude spectrograms for the majority of the experiments in this thesis. This last section includes an attempt that aims at generating an estimate of the target phase which is more precise than using the mixture's phase as an estimate.

4.9.1 Phase estimation with self-attention

A representation of the architecture used in the experiment is presented in figure 4.8. In this experiment we try to exploit the Transformer Encoder to generate a better estimate of the phase information. The model is divided into two branches, namely the magnitude and phase branch. As the names suggest, one acts on the magnitude spectrogram and one on the phase spectrogram. The first one is an exact replica of the model in figure 4.2 and outputs an estimate using masking applied on the magnitude spectrogram. The phase branch, on the other hand, starts from the mixture's phase, applies positional encoding to it and then passes the phase spectrogram through a 2 layers deep Transformer Encoder with 5 heads a feed forward with hidden dimension equal to 2048. As opposed to the spectrogram branch, here the output of the Encoder is not a mask but rather a direct estimate of the target's phase. Both the magnitude and phase estimates are used to compute the loss, which is defined as a combination of their MAE losses, namely:

$$Loss = MAE(\hat{S} - S) - MAE(\hat{P} - P) \tag{4.2}$$

where S and P represent the target's magnitude and phase spectrograms and $\hat{S}$ and $\hat{P}$ their estimates generate with our model. Results are not great, as shown in table 4.11.

	Stem			
Architecture	Vocals	Bass	Drums	Other
Baseline	**4.914**	**3.299**	**4.660**	**0.984**
PhaseEstimation	1.302	0.772	0.656	0.402

Table 4.11: Median SDR on the test set for the phase estimation architecture against the baseline

We can notice that there is a drop in performance for all 4 stems with a minimum decrease of 0.58dB in median SDR for *other* and a maximum drop of 4.01dB in the same metric for *drums*. Given the poor results obtained in this experiments and the struggles encountered in the preliminary ones, we decided to stop the research in the phase estimation field since the task seems too difficult for our models.

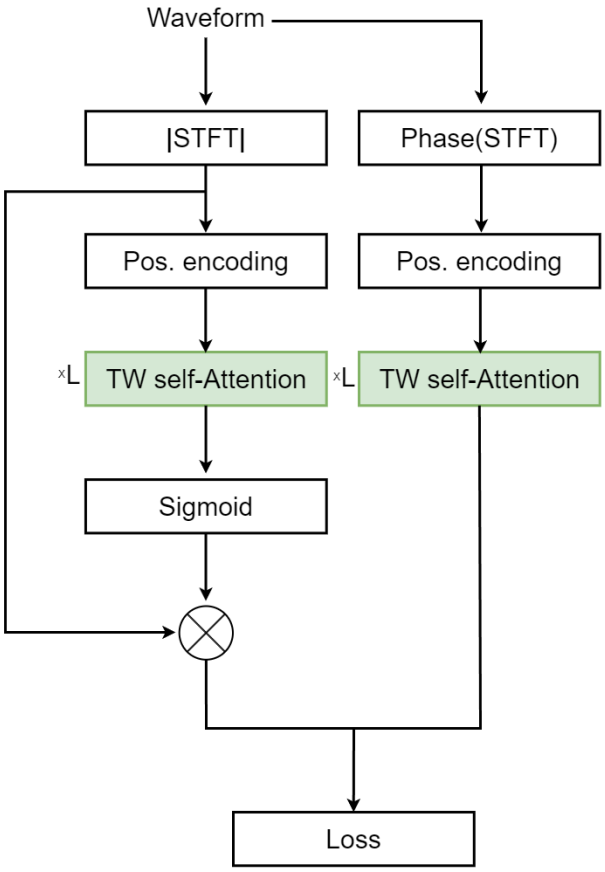

Figure 4.8: Model architecture for the phase experiment

Chapter 5

Conclusions

In this section we will analyze the results presented in the Experiments chapter of this thesis and draw conclusions on the effect of Multi-Head Attention in MSS.

5.1 Base experiment

The incorporation of self-attention algorithms into the framework of Music Source Separation demonstrates promising results. While the achieved outcomes fall short of state-of-the-art performances, the performance gap seems to become smaller and smaller as the number of training samples increases. It should also be noted that the additional data used for training have been automatically labelled using HT Demucs and the fine-tuned version of HT Demucs, both of which offer suboptimal estimates for the target. SOTA models like the ones just mentioned have been trained on a dataset that is 2.5 times bigger than the one used in this work. Combining this quantitative advantage with the qualitative one we just mentioned puts our model at a disadvantage. While the results obtained with a bigger dataset may be promising, we can also notice that the model exploiting FW Attention can not compete with other architectures when trained on MUSDB18 only and this is probably due to the nature of the Multi-Head Attention algorithm, which is notoriously a data hungry one. The experiments highlighted a preference of MSS models for shallower architectures, preference that finds confirmation in the fact that most SOTA models implement either one or two sequential modelling units [59][52][66]. In our experiment, 2 Transformer Encoder layers resulted in the most accurate separations. The comprehensive ablation study conducted on the primary architecture has yielded meaningful insights into Source Activity Detection (SAD) and the incorporation of Scale-Invariant Signal-to-Distortion Ratio (SI-SDR) based loss functions. The integration of SAD in the data processing pipeline results in noteworthy improvements and the SAD implemented in [52] offers a good balance

between salient and silent audio chunks during training, contributing to a more robust framework. On the SI-SDR losses, our findings suggest a potential drawback linked to the redundancy of information provided by SI-SDR, which apparently does not complement well the MAE computed on the magnitude spectrogram. Moreover, due to how the SI-SDR is formulated, it is easier to achieve very low SDR values for the samples corresponding to silent audio chunk, thus weighting them more than their salient counterparts. This imbalance in the weights contrasts the positive effects of SAD. The conclusion we draw is that for our experiments the MAE loss computed on the magnitude spectrogram is the optimal one as it does not modify the balance between salient and silent chunks obtained through to the SAD procedure. As expected, the exploration of larger batch sizes and the expansion of the validation set have demonstrated noteworthy improvements in the median SDR of our MSS model. Increasing the batch size has facilitated more efficient parameter updates during training and the expansion of the validation set has provided a more comprehensive evaluation of the model's generalization capabilities, ensuring that it performs robustly across a broader spectrum of data. While the observed enhancements align with established principles in machine learning, the predictability of these outcomes underscores the reliability and consistency of the implemented strategies. On a more general note, in the course of our MSS experiment, it became evident that predicting the stem *vocals* posed a relatively more straightforward challenge compared to the drums and bass components. The distinct vocal range inherent in the stem vocals facilitated a clearer pattern recognition, allowing the model to capture and segregate vocal elements more accurately. On the other hand, *drums* and *bass* presented intermediate levels of complexity, with *drums* usually obtaining the best performance between the two. However, the most noteworthy observation lays in the unpredictable nature of the stem labeled as 'other.' This stem is inherently diverse, incorporating a wide array of sound sources that defy easy categorization. The complexity arises from the merging of various instruments and ambient elements within this stem, making it challenging for the model to accurately predict and separate these diverse sonic components. Our experiments show that this intrinsic diversity can be partially tackled by the use of a larger batch size and an expansion of the validation set.

We think that an interesting direction for future work on the subject could include training on a higher quality and higher quantity dataset with more architectural fine-tuning, e.g. trying different positionings of the layer norm blocks, introducing convolutions to make the Encoder more similar to a Conformer and testing with different dimensions of the feed-forward layer. Moreover, using a MLP for mask estimation could be beneficial[52] and should be kept into consideration. The current SOTA model trained on magnitude spectrograms is D3Net and we believe that with these adjustments in the model's architecture and in the training data its performance could be matched.

5.2 Impact of FW Attention

While the introduction of a FW Attention block seemed to help in the separation of *other*, the presence of this block seems to confuse the model in the separation of the other three stems since it is not able to replicate the results obtained with the baseline. Between the parallel and consecutive architectures, the consecutive one appears to be the best performing one. Future work on the subject should focus on architectural modifications with a focus on the implementation of more complex architectures such as the ones presented in Figure 3 in [61].

5.3 SSL features introduction

Despite optimistic expectations powered by the work in [28], the incorporation of self-supervised features from the last encoder layer of WavLM Base+ into our Music Source Separation model did not yield the anticipated improvements in performance. The intention behind leveraging features from the final encoder layer was to capture high-level representations that encapsulate complex dependencies within the audio signals. However, the observed outcomes indicate that extracting features solely from the last encoder layer did not effectively enhance the model's ability to separate sources. This suggests a potential mismatch between the extracted self-supervised features and the structure present in music data, leading to limited improvements in source separation accuracy and, for some stems, even a decrease in performance. We think that this result comes from the mismatch between MUSDB18 and the dataset used for training WavLM, which is a mixture of Libri-Light[82], GigaSpeech[83] and VoxPopuli[84], i.e. speech oriented datasets.
A notable shift in outcomes was observed when self-supervised features were integrated in our Music Source Separation model as a combination of all layer representations. This more versatile approach demonstrated a discernible improvement for *vocals* and *bass* performance. By aggregating information from multiple layers, the model exhibited a greater capacity to capture diverse patterns within the audio signals, encouraging a deeper understanding of the underlying structure of the stem. The success of this integrated approach suggests that a comprehensive utilization of self-supervised features that exploits information from all hierarchical levels can contribute to the model's ability to perform Music Source Separation effectively.
Upon scrutinizing the learned weights associated with each layer in our Music Source Separation model, an interesting pattern emerged, revealing that the last four layers consistently held more importance for all four stems—*vocals*, *drums*, *bass*, and *other*. These final layers played an important role in capturing waveform dependencies and improving the model's understanding of the audio signal. Intriguingly, in the case of

the *vocals* stem, a departure from this trend was observed, as the initial layers were associated with non-negligible weights. This suggests that the earliest layers played a more significant role in encoding relevant features for vocal source separation. In stark contrast, for the *drums*, *bass*, and *other* stems, the weights associated with the first nine layers were found to be almost 0, indicating that the deeper layers were predominantly responsible for capturing the essential characteristics of the input waveforms.

Substituting WavLM Base+ with HuBERT and Wav2Vec2 trained on Audioset yielded encouraging results. Both HuBERT models helped improving the performance. They achieved the same average median SDR across the 4 metrics, which is 0.14dB higher than the one obtained with WavLM Base+. The best performing model has been Wav2Vec2 Base, while the worst one has been Wav2Vec2 Large. Using general purpose models which have been trained on a more diverse dataset yields encouraging results on the MSS downstream task, reinforcing the results obtained for audio classification tasks in [62].

The findings from our Music Source Separation research underscore the potential for future work to pivot towards the development of a general pre-trained model tailored for music processing tasks. A critical aspect of this work would involve refining the model's architecture and training parameters to accommodate a more suitable nFFT (number of samples in each frame) parameter, such as 2048. This adjustment should help in dealing with the peculiar characteristics of music signals, allowing the model to capture finer spectral features within the audio data.

Establishing a pre-trained model for music with optimized parameters would promote transfer learning across a variety of music-related tasks, including source separation, genre classification, and music generation. This approach would lay the foundation for a more comprehensive understanding and utilization of deep learning features in the MSS domain.

5.4 RMSE and ZCR introduction

Concatenating the ZCR to the model seems to boost the performance for *vocals* and *bass*, while lowering the one for *drums* and *other*. The introduction of the RMSE features seems to output more consistent results, increasing the performance of the reference baseline on all 4 stem predictions. The concatenation of both features seems to be effective only for *bass* estimation, while for the other three stems either ZCR or RMSE should be preferred.

5.5 Phase estimation attempts

Despite the advancements made in our Music Source Separation model, it is crucial to acknowledge the challenges encountered in the phase estimation task. Our attempts to estimate robust phase information into the separation process proved to be harmful to the model's performance, suggesting that this task remains too difficult even for a Multi-Head Attention mechanism. The complex nature of phase estimation, crucial for reconstructing the temporal alignment and coherence of audio signals, makes it a challenge that extends beyond the capabilities of the Multi-Head Attention. Future work in this domain should focus on refining methodologies or exploring alternative techniques specifically designed to tackle the task of accurate phase estimation in the context of music source separation.

Appendix A

Proof of equation 2.3

This appendix contains the proof of the following equation:

$$\int_{-\pi}^{\pi} e^{inx} e^{-imx} dx = \begin{cases} 0, & \text{if } n \neq m \\ 2\pi, & \text{if} n = m \end{cases}$$

Starting from the integral, we can group together the two terms

$$\int_{-\pi}^{\pi} e^{inx} e^{-imx} dx = \int_{-\pi}^{\pi} e^{i(n-m)x} dx$$

When $n = m$:

$$\int_{-\pi}^{\pi} e^{i0x} dx = \int_{-\pi}^{\pi} 1 dx = 2\pi$$

When $n \neq m$, by setting $t = (n - m)$ we obtain:

$$\int_{-\pi}^{\pi} e^{itx} dx = \frac{1}{it}(e^{it\pi} - e^{-it\pi})$$

And we can further simplify the equation by exploiting Euler's formula:

$$\int_{-\pi}^{\pi} e^{itx} dx = \frac{1}{it}[cos(t\pi) + isin(t\pi) - cos(-t\pi) - isin(-t\pi)]$$

Since $cos(t\pi) = cos(-t\pi)$ and $sin(t\pi) = -sin(t\pi) = 0$, when $n \neq m$ the integral is equal to 0 and equation 2.3 holds.

Appendix B

Proof of conjugate symmetricity of Fourier Coefficients

This appendix offers a proof that the series of complex coefficients c_n is conjugate symmetric, meaning that $c_n = \overline{c_{-n}}$. Starting from the formula for the complex coefficients:

$$c_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-inx}dx$$

The formula for the opposite index coefficients will be:

$$c_{-n} = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{inx}dx$$

And by applying the conjugate operation on both sides:

$$\overline{c_{-n}} = \frac{1}{2\pi}\overline{\int_{-\pi}^{\pi} f(x)e^{inx}dx} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \overline{f(x)}\overline{e^{inx}}\overline{dx}$$

But since x, f and d are real valued, the conjugate of a real valued function evaluted at x is equal to the function evaluated at x. Adding the fact that $\overline{e^{inx}} = e^{-inx}$, this means that the following equation holds:

$$\overline{c_{-n}} = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-inx}dx = c_n$$

Since the amplitude of a complex coefficient is equal to its squared real and imaginary parts, it follows that conjugate symmetric coefficients must also have the same amplitude.

Appendix C

Training Times

In this appendix we report the training times for every configuration tested. It should be noted that the architectures including SSL features and ZCR and RMSE features have been trained on a the HPC cluster[73], whhile all the other experiments have been carried out on a nVidia GeForce GTX 1650 GPU with 4GB of memory. On this GPU we also carried out preliminary tests for the SSL configuration and training proceded at roughly 35 seconds per iteration. On the HPC cluster, the training speed was 2.5 iterations per seconds or 0.4 seconds per iteration, which is a 87.5x faster. This means that training the SSL architectures on the same computational setting as all the other experiments would have taken roughly 29 days. This is due to the computationally expensive operation of extracting the self supervised features with WavLM Base+.

	Stems				
Architecture	Vocals	Bass	Drums	Other	Total
Baseline L=6	00:12:42	00:15:19	00:15:28	00:17:22	01:00:51
Baseline L=2	00:08:06	00:08:14	00:07:38	00:03:23	00:27:21
Baseline L = 1	00:03:46	00:06:27	00:03:44	00:03:15	00:17:12
L=2 + 260 songs	00:22:22	00:22:58	00:30:32	00:11:55	01:27:47
BS=128	01:17:14	01:45:18	01:18:16	00:41:30	05:02:18
Modified Loss	00:12:37	00:17:17	00:09:11	00:10:28	00:49:33
Without SAD	00:32:18	00:24:25	00:20:57	00:11:50	01:29:30
Parallel TW-FW	00:28:59	00:26:15	00:57:51	00:43:18	02:36:23
Consec. TW-FW	00:34:47	00:28:51	00:41:21	01:07:39	02:52:38
Phase estimation	00:09:41	00:33:49	00:10:02	00:12:27	01:05:59

Table C.1: Training times in hh:mm:ss format for architectures trained on nVidia GeForce GTX 1650

	Stems				
Architecture	Vocals	Bass	Drums	Other	Total
RMSE	00:19:10	00:09:34	00:23:00	00:22:22	01:14:06
ZCR	00:25:00	00:23:47	00:07:06	00:11:55	01:07:48
RMSE and ZCR	00:35:27	00:27:16	00:37:51	00:10:49	01:51:23
WavLM Base+ (last layer)	01:37:53	01:03:52	03:18:33	02:04:34	08:04:52
WavLM Base+ (hid. layers)	01:23:38	01:48:32	02:15:26	02:33:22	08:00:58
HuBERT Base - Audioset	02:13:42	01:39:14	03:06:12	01:56:03	08:55:11
HuBERT Large - Audioset	02:16:56	02:46:58	04:19:33	04:10:28	13:33:55
Wav2Vec2 Base - Audioset	01:49:10	00:42:37	01:43:26	01:56:41	06:11:54
Wav2Vec2 Large - Audioset	03:10:33	03:18:22	05:30:46	04:40:03	16:39:44

Table C.2: Training times in hh:mm:ss format for architectures trained on the HPC cluster

Appendix D

SDR distribution for *bass*, *drums* and *other*

In this appendix we present the SDR distribution for *bass*, *drums* and *other*. We can observe that *bass* and *drums* follow distribution which is very similar distribution to the one in figure 4.3, with a higher values of the SDR associated with higher values of the ration between the RMSEs of the target and mixture chunks. The model outputs unsuccessfull predictions when the target chunk is silent and the mixture is not. The stem *other* features a more normal-like distribution. This is probably caused by the fact that this stem is rarely silent as table 4.1 shows. For this reason we do not have two noticeable peaks in the distribution plot for this stem, although if we look at the values of the SDR where the ratio is equal to 0 we can notice that they show similar behaviour to the ones observed in the other stems. As a last remark we explain the fact that the *drums* stem presents values of the ratio that are higher than 1. This is due to an error in MUSDB18 which is reported in their website [65]. The track 'Oh No' by PR reportedly has the following issue "sum of sources does not add up to the mix for the left channel" [65]. Since we averaged the two channels to generate a mono channel audio, the same holds for our modified dataset and in this very track there is a section when only the drums are present and for this reason the ratio becomes bigger than 1.

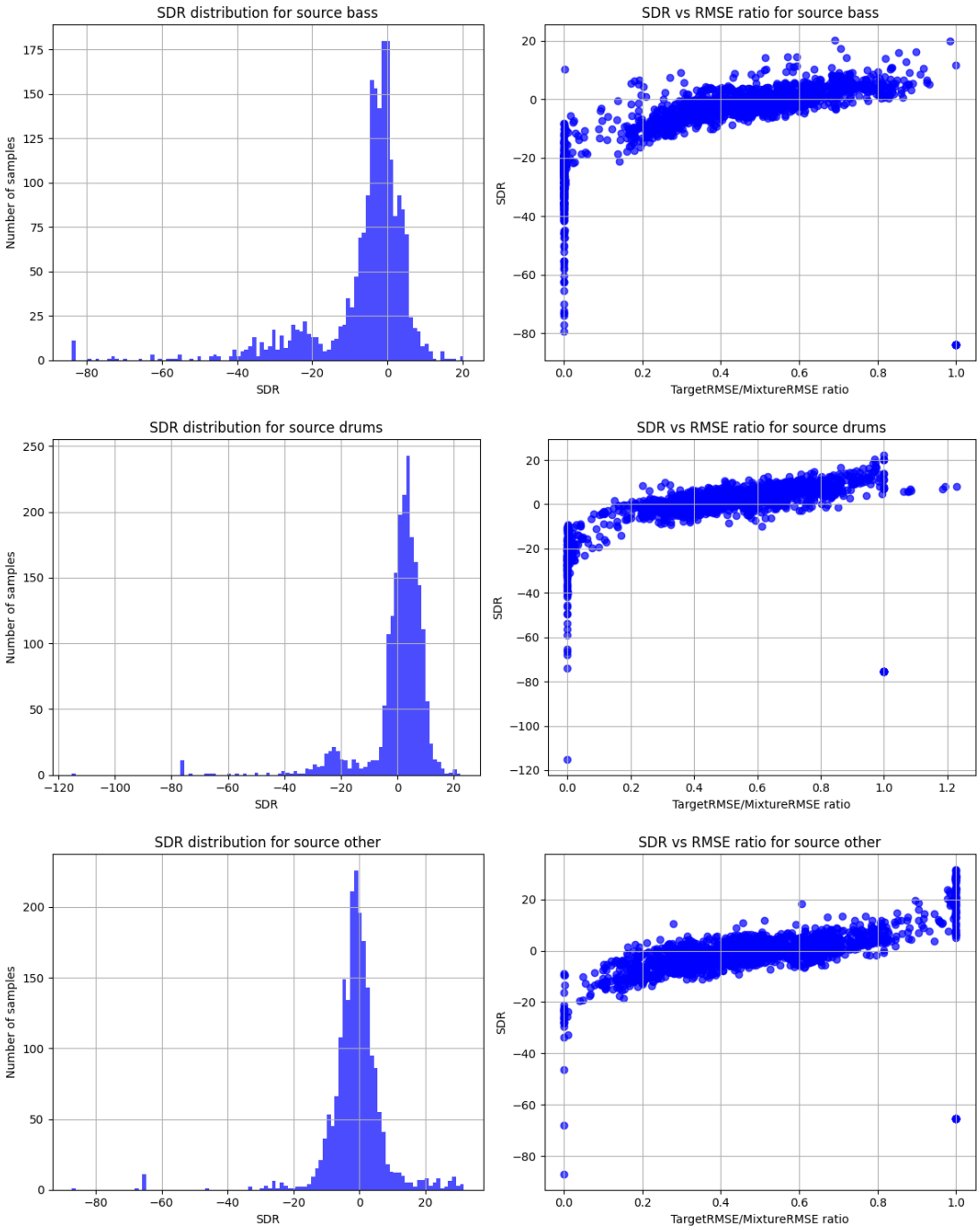

Figure D.1: SDR distribution for the source not presented in figure 4.3

Appendix E

Inference speed

In this appendix, we present the inference times for the tested models, showcasing their performance on a randomly generated dataset. The dataset is composed of 100 waves, each consisting of 2.88 million samples, simulating songs with a duration of 3 minutes and sampled at 16kHz. This dataset results in a database containing 5 hours of simulated music and the inference speeds of the models have been measured using this database. The recorded inference times are reported in table E.1, illustrating the duration required for completing the whole MSS pipeline. This includes the time taken to perform Short-Time Fourier Transform (STFT) on the initial waveform, generate a magnitude spectrogram estimate using our model, and execute inverse STFT (iSTFT) on the prediction. The reported results are the average across 10 repetitions of the test and specifically pertain to the time required to process a single stem. The GPU used is a nVidia GeForce GTX 1650 while the CPU is a AMD Ryzen 5 5600H. All experiments are reported except the ones for which no architectural change was required. For example, the experiment using the SISDR loss only modifies the training loss and should therefore have the same inference speed as the model in the second row of table E.1.

Architecture	**GPU Time**	**CPU Time**
Baseline L = 6	32.210	356.140
Baseline L = 2	12.116	121.902
Baseline L = 1	7.332	61.934
Parallel TW-FW	21.929	290.924
Consec. TW-FW	23.381	348.220
Phase estimation	22.711	305.119
RMSE	11.879	109.802
ZCR	11.874	110.691
RMSE and ZCR	12.962	123.997
WavLM Base+ (last layer)	339.362	4170.005
WavLM Base+ (hid. layers)	337.143	3550.011
HuBERT Base - Audioset	314.761	3485.348
HuBERT Large - Audioset	885.268	11694.390
Wav2Vec2 Base - Audioset	319.652	3550.011
Wav2Vec2 Large - Audioset	872.592	10742.480

Table E.1: Average Inference time tested on GPU and CPU over 10 iterations. Results are expressed in seconds.

Bibliography

[1] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015 (cit. on pp. 4, 20).

[2] Meinard Muller. «Fundamentals of Music Processing». In: Springer Cham, 2015. Chap. 1 (cit. on p. 6).

[3] Zafar Rafii et al. *MUSDB18-HQ - an uncompressed version of MUSDB18*. Aug. 2019 (cit. on pp. 17, 18, 28).

[4] Frank Rosenblatt. «The perceptron: a probabilistic model for information storage and organization in the brain.» In: *Psychological review* 65 6 (1958), pp. 386–408. URL: https://api.semanticscholar.org/CorpusID:12781225 (cit. on p. 18).

[5] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. «Multilayer feedforward networks are universal approximators». In: *Neural Networks* 2.5 (1989), pp. 359–366 (cit. on p. 19).

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016 (cit. on p. 19).

[7] Hugo Touvron et al. *ResMLP: Feedforward networks for image classification with data-efficient training*. 2021 (cit. on p. 19).

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. «Learning Internal Representations by Error Propagation». In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press, 1986, pp. 318–362 (cit. on p. 19).

[9] Yi Luo and Jianwei Yu. *Music Source Separation with Band-split RNN*. 2022 (cit. on p. 19).

[10] LeCun et al. «Backpropagation Applied to Handwritten Zip Code Recognition». In: *Neural Computation* 1.4 (1989), pp. 541–551 (cit. on p. 19).

[11] David H. Hubel and Torsten N. Wiesel. *Brain and Visual Perception: The Story of a 25-year Collaboration*. Oxford University Press, 2004 (cit. on p. 19).

[12] Kunihiko Fukushima. «Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position». In: *Biological Cybernetics* 36 (1980), pp. 193–202 (cit. on p. 19).

[13] Shuying Liu and Weihong Deng. «Very deep convolutional neural network based image classification using small training sample size». In: *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (2015), pp. 730–734 (cit. on p. 20).

[14] R. Tamilarasi and S. Gopinathan. «Inception Architecture for Brain Image Classification». In: *Journal of Physics: Conference Series* 1964.7 (2021), p. 072022 (cit. on p. 20).

[15] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 20).

[16] Felix Gers, Jürgen Schmidhuber, and Fred Cummins. «Learning to Forget: Continual Prediction with LSTM». In: *Neural computation* 12 (Oct. 2000), pp. 2451–71 (cit. on p. 20).

[17] Matthew E. Peters et al. *Deep contextualized word representations*. 2018 (cit. on p. 20).

[18] Alex Graves et al. «Biologically Plausible Speech Recognition with LSTM Neural Nets». In: *Biologically Inspired Approaches to Advanced Information Technology*. Springer Berlin Heidelberg, 2004 (cit. on p. 20).

[19] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. «Speech recognition with deep recurrent neural networks». In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), pp. 6645–6649 (cit. on p. 20).

[20] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016 (cit. on p. 20).

[21] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014 (cit. on p. 20).

[22] Alex Graves and Jürgen Schmidhuber. «Framewise phoneme classification with bidirectional LSTM and other neural network architectures». In: *Neural Networks* 18.5 (2005), pp. 602–610 (cit. on p. 20).

[23] Ashish Vaswani et al. *Attention Is All You Need*. 2023 (cit. on pp. 20, 21).

[24] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016 (cit. on p. 21).

[25] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019 (cit. on pp. 22, 23, 30).

[26] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020 (cit. on pp. 22, 23).

[27] Steffen Schneider et al. *wav2vec: Unsupervised Pre-training for Speech Recognition*. 2019 (cit. on pp. 23, 30).

[28] Sanyuan et al. Chen. «WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing». In: *IEEE Journal of Selected Topics in Signal Processing* 16.6 (2022), pp. 1505–1518 (cit. on pp. 23, 30, 31, 52, 61).

[29] Wei-Ning Hsu et al. *HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units*. 2021 (cit. on pp. 23, 30).

[30] Giorgio et al. Fabbro. *The Sound Demixing Challenge 2023 - Music Demixing Track*. 2024 (cit. on pp. 24, 32, 33).

[31] *Ultimate Vocal Remover, gitHub*. https://github.com/Anjok07/ultimatevocalremovergui (cit. on p. 24).

[32] Peter J. Huber. «Projection Pursuit». In: *The Annals of Statistics* 13.2 (1985), pp. 435–475 (cit. on p. 25).

[33] M. C. Jones and Robin Sibson. «What is Projection Pursuit?» In: *Journal of the Royal Statistical Society. Series A (General)* 150.1 (1987), pp. 1–37 (cit. on p. 25).

[34] A. Hyvärinen and E. Oja. «Independent component analysis: algorithms and applications». In: *Neural Networks* 13.4 (2000), pp. 411–430 (cit. on p. 25).

[35] Daniel D. Lee and H. Sebastian Seung. «Learning the parts of objects by nonnegative matrix factorization». In: *Nature* 401 (1999), pp. 788–791 (cit. on p. 25).

[36] E. Wachsmuth, M. W. Oram, and D. I. Perrett. «Recognition of Objects and Their Component Parts: Responses of Single Units in the Temporal Cortex of the Macaque». In: *Cerebral Cortex* 4.5 (Sept. 1994), pp. 509–522 (cit. on p. 25).

[37] N. K. Logothetis and D. L. Sheinberg. «Visual object recognition». In: *Annual review of neuroscience* 19 (1996), pp. 577–621 (cit. on p. 25).

[38] Paris Smaragdis. «Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs». In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer, 2004, pp. 494–499 (cit. on p. 25).

[39] Paris Smaragdis and Judith C. Brown. «Non-negative matrix factorization for polyphonic music transcription». In: *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (2003), pp. 177–180 (cit. on p. 25).

[40] Shankar Vembu and Stephan Baumann. «Separation of Vocals from Polyphonic Audio Recordings .» In: Jan. 2005, pp. 337–344 (cit. on p. 25).

[41] Luis Gustavo Martins et al. «Polyphonic Instrument Recognition Using Spectral Clustering». In: *Proceedings of the 8th International Conference on Music Information Retrieval*. Austrian Computer Society, 2007, pp. 213–218 (cit. on p. 25).

[42] Mathieu Lagrange et al. «Normalized Cuts for Predominant Melodic Source Separation». In: *Audio, Speech, and Language Processing, IEEE Transactions on* 16 (2008), pp. 278–290 (cit. on p. 26).

[43] Mathieu Lagrange and George Tzanetakis. «Sound Source Tracking and Formation using Normalized Cuts». In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol. 1. 2007, pp. I-61-I–64 (cit. on p. 26).

[44] Yiming Lin et al. «Unsupervised Harmonic Sound Source Separation with Spectral Clustering». 2021 (cit. on p. 26).

[45] Frederick Yen, Mao-Chang Huang, and Tai-Shih Chi. «A two-stage singing voice separation algorithm using spectro-temporal modulation features». In: 2015, pp. 3321–3324 (cit. on p. 26).

[46] A.M. Ahmad et al. «An isolated speech endpoint detector using multiple speech features». In: *2004 IEEE Region 10 Conference*. Vol. B. 2004, 403–406 Vol. 2 (cit. on p. 26).

[47] Y. Qi and B.R. Hunt. «Voiced-unvoiced-silence classifications of speech using hybrid features and a network classifier». In: *IEEE Transactions on Speech and Audio Processing* 1.2 (1993), pp. 250–255 (cit. on p. 26).

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015 (cit. on p. 26).

[49] A. Jansson et al. «Singing voice separation with deep U-Net convolutional networks». Unpublished. 2017 (cit. on pp. 26, 27).

[50] Fabian-Robert Stöter et al. «Open-Unmix - A Reference Implementation for Music Source Separation». In: *Journal of Open Source Software* 4.41 (2019), p. 1667 (cit. on p. 27).

[51] Naoya Takahashi and Yuki Mitsufuji. *D3Net: Densely connected multidilated DenseNet for music source separation*. 2021 (cit. on pp. 28, 40).

[52] Yi Luo and Jianwei Yu. *Music Source Separation with Band-split RNN*. 2022 (cit. on pp. 28, 35, 38, 40, 43, 48, 49, 51, 59, 60).

[53] Woosung Choi et al. *Investigating U-Nets with various Intermediate Blocks for Spectrogram-based Singing Voice Separation*. 2020 (cit. on p. 28).

[54] Francesc Lluís, Jordi Pons, and Xavier Serra. *End-to-end music source separation: is it possible in the waveform domain?* 2019 (cit. on p. 28).

[55] Daniel Stoller, Sebastian Ewert, and Simon Dixon. *Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation*. 2018 (cit. on p. 29).

[56] Alexandre Défossez et al. *SING: Symbol-to-Instrument Neural Generator*. 2018 (cit. on p. 29).

[57] Alexandre Défossez et al. *Music Source Separation in the Waveform Domain*. 2021 (cit. on p. 29).

[58] Minseok Kim et al. *KUIELab-MDX-Net: A Two-Stream Neural Network for Music Demixing*. 2021 (cit. on p. 29).

[59] Simon Rouard, Francisco Massa, and Alexandre Défossez. *Hybrid Transformers for Music Source Separation*. 2022 (cit. on pp. 30, 37, 38, 43, 46, 48, 59).

[60] Anmol Gulati et al. *Conformer: Convolution-augmented Transformer for Speech Recognition*. 2020 (cit. on p. 30).

[61] Yunkee Chae et al. *Exploiting Time-Frequency Conformers for Music Audio Enhancement*. 2023 (cit. on pp. 30, 51, 61).

[62] Moreno La Quatra et al. «Benchmarking Representations for Speech, Music, and Acoustic Events». In: *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops*. 2024 (cit. on pp. 31, 62).

[63] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. «Performance measurement in blind audio source separation». In: *IEEE Transactions on Audio, Speech and Language Processing* 14.4 (2006), pp. 1462–1469 (cit. on p. 31).

[64] Jonathan Le Roux et al. *SDR - half-baked or well done?* 2018 (cit. on pp. 32–34).

[65] Zafar Rafii et al. *The MUSDB18 corpus for music separation*. Dec. 2017 (cit. on pp. 35, 36, 68).

[66] Alexandre Défossez. *Hybrid Spectrogram and Waveform Source Separation*. 2022 (cit. on pp. 35, 43, 59).

[67] Minseok Kim, Jun Hyung Lee, and Soonyoung Jung. *Sound Demixing Challenge 2023 Music Demixing Track Technical Report: TFC-TDF-UNet v3*. 2023 (cit. on p. 35).

[68] Rachel M. Bittner et al. «MedleyDB 2.0: New Data and a System for Sustainable Data Collection». In: 2016 (cit. on p. 37).

[69] Rachel Bittner et al. «MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research». In: 2014 (cit. on p. 37).

[70] *MedlyDB's gitHub page*. https://github.com/marl/medleydb (cit. on p. 37).

[71] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021 (cit. on pp. 37, 45).

[72] Brian et al. McFee. *librosa/librosa: 0.10.0.post2*. Zenodo, 2023. URL: `https://zenodo.org/record/7746972` (cit. on p. 38).

[73] *Cluster HPC, Politecnico di Torino*. http://hpc.polito.it (cit. on pp. 39, 66).

[74] Ashish Vaswani et al. *Attention Is All You Need*. 2023 (cit. on pp. 39, 41, 42).

[75] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. *The 2018 Signal Separation Evaluation Campaign*. 2018 (cit. on p. 40).

[76] Romain Hennequin et al. «Spleeter: a fast and efficient music source separation tool with pre-trained models». In: *Journal of Open Source Software* 5 (2020), p. 2154 (cit. on p. 40).

[77] Thomas Sgouros, Angelos Bousis, and Nikolaos Mitianoudis. «An Efficient Short-Time Discrete Cosine Transform and Attentive MultiResUNet Framework for Music Source Separation». In: *IEEE Access* 10 (2022), pp. 119448–119459 (cit. on p. 43).

[78] Ryosuke Sawata et al. *The Whole Is Greater than the Sum of Its Parts: Improving DNN-based Music Source Separation*. 2023 (cit. on p. 43).

[79] Zili Huang et al. *Investigating self-supervised learning for speech enhancement and separation*. 2022 (cit. on p. 52).

[80] Xiangning Chen et al. *Symbolic Discovery of Optimization Algorithms*. 2023 (cit. on p. 56).

[81] Pierre Foret et al. *Sharpness-Aware Minimization for Efficiently Improving Generalization*. 2021 (cit. on p. 56).

[82] J. Kahn et al. «Libri-Light: A Benchmark for ASR with Limited or No Supervision». In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2020 (cit. on p. 61).

[83] Guoguo Chen et al. *GigaSpeech: An Evolving, Multi-domain ASR Corpus with 10,000 Hours of Transcribed Audio*. 2021 (cit. on p. 61).

[84] Changhan Wang et al. *VoxPopuli: A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation*. 2021 (cit. on p. 61).