Technische Universität Berlin

Audio Communication Group

Fakultät I
Einsteinufer 17
10587 Berlin
http://www.ak.tu-berlin.de

Master Thesis

Adding Context Information to Deep Neural Network based Audio Source Separation

Ramón Wiegratz - ramon.wiegratz@campus.tu-berlin.de - AKT - Mat. -Nr. 395920

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt gegenüber der Fakultät I der Technischen Universität Berlin, dass die vorliegende, dieser Erklärung angefügte Arbeit selbstständig und nur unter Zuhilfenahme der im Literaturverzeichnis genannten Quellen und Hilfsmittel angefertigt wurde. Alle Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, sind kenntlich gemacht. Ich reiche die Arbeit erstmals als Prüfungsleistung ein. Ich versichere, dass diese Arbeit oder wesentliche Teile dieser Arbeit nicht bereits dem Leistungserwerb in einer anderen Lehrveranstaltung zugrunde lagen.

Titel der schrifltichen Arbeit

Adding Context Information to Deep Neural Network based Audio Source Separation

Verfasser

Ramón Joscha Wiegratz (395920)

Betreuende Dozenten

Prof. Dr. Stefan Weinzierl
Athanasios Lykartsis
Roman Gebhardt

Mit meiner Unterschrift bestätige ich, dass ich über fachübliche Zitierregeln unterrichtet worden bin und verstanden habe. Die im betroffenen Fachgebiet üblichen Zitiervorschriften sind eingehalten worden. Eine Überprüfung der Arbeit auf Plagiate mit hilfe elektronischer Hilfsmittel darf vorgenommen werden.

Berlin, den 07. Februar 2021

Abstract

This thesis deals with deep learning-based music source separation. Based on a convolutional time-frequency masking architecture, a number of unique strategies are developed that enhance the receptive field of the network to a more relevant and music-specific context. This leads to a better exploitation of the data, as well as a more stable training convergence. Firstly, the exponentially scaled harmonic relationships inside a spectrogram are made accessible to the convolutional neural network by restructuring the input data and aligning harmonically related frequency bins over the feature dimension. Secondly, a post-processing network is proposed to re-align the contaminated mixture phase, which is commonly used for the reconstruction of the source signals. Finally, a novel self-attention mechanism is proposed, which is able to filter out unwanted interferences and distortions by utilizing the repetitive nature of music. All proposed methods are evaluated on the MusDB18 dataset and improve the instrument separation. The final separation method outperforms the current State-of-the-Art results on MusDB18. Additionally, an ablation study on different source separation network architectures and training methods is provided with this thesis.

Zusammenfassung

Diese Masterarbeit behandelt die Quellen Separierung von Musik mithilfe von Deep Learning. Grundlage bildet ein Convolutional Neural Network (CNN) welches einen spektralen Filter berechnet. Hierfür werden spezielle Methoden entwickelt, mit welchen das Netzwerk relevante und Musik-spezifische Informationen besser Nutzen kann. Diese führen zu einer effizienteren Datenauswertung und besserer Lern-Convergence. Zunächst werden die exponentiell skalierten harmonischen Zusammenhänge innerhalb eines Spektrogramms dem CNN zugänglicher gemacht, indem das Eingangs-Spektrogramm umstrukturiert wird und harmonisch verwandte Frequenzbereiche über die Feature Dimension aufgereiht werden. Des Weiteren wird ein Post-Processing CNN genutzt, um die verrauschte Mix Phase, die üblicherweise für die Rekonstruktion der Quell-Signale verwendet wird, anzugleichen. Schließlich wird ein neuer self-Attention Mechanismus vorgeschlagen, der unerwünschte Interferenzen und Verzerrungen herausfiltert, indem die repetitiven Eigenschaften von Musik ausgenutzt werden. Alle vorgeschlagenen Methoden verbessern die Separierung von Instrumenten auf dem MusDB18-Datensatz. Die finale Methode übertrifft die aktuellen State-of-the-Art Ergebnisse, evaluiert auf MusDB18. Darüber hinaus wird in dieser Arbeit eine Studie über verschiedene Netzwerk Architekturen und Trainingsmethoden zur Quellen Separierung durchgeführt.

Acknowledgements

I would like to thank the following people who all helped me, each in their own way, to finish this thesis.

Athanasios Lykartsis for his constant support, trust and for giving me the time and freedom to find my own approach to this topic.

Roman Gebhardt for helping me during this thesis and for the many excellent seminars, which really encouraged me during my studies.

Thomas Koch and **Christian Weißig** from Fraunhofer HHI, for giving me complete access to a server with two NVIDIA GTX 1080 Ti, which I was able to use throughout this thesis. Without this computational power I wouldn't have been able to achieve my results.

Anvit and **Sanju** for helping me revising the written part of this thesis.

And last but not least, **my parents** for their unconditional love and always being supportive.

Thank you!

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation and Goals

After 50 years of research, Music Source Separation has reached a stage, where first commercial application are beginning to emerge. The ability to separate single source instruments out of a song has many different applications. Besides the obvious benefits for Karaoke, musicians will be able to cover songs much easier, as specific play-alongs can be easily created. Djs will have much more sophisticated ways of mixing and remixing different songs together. Sampling, the art of creating new music out of already existing music, can be used in much more precise and unrestricted ways, although copyright infringements might become harder to detect. Also other scientific research fields, especially the Music Information Retrieval (MIR), will benefit, as analytic tasks, like automatic music transcription, will become more straightforward. Source separation could lead to massive amounts of musical data that, for example, could be used to train new generative models, like WaveNets, on music synthesis.

Currently, the design of music source separation models is mostly influenced by recent developments in the field of deep learning. Hence, most research is focused on the adaption of general deep learning methods to solve the music source separation problem, while often not taking into account the specific characteristics and properties of music. However, previous approaches, which were not based on deep learning, have already shown that these specific properties can be utilized to achieve separation. In this thesis the current deep learning methods are examined and their shortcomings are pointed out. Thereafter, solutions to the identified problems are developed based on specific properties of music, like the harmonic series and structural repetition. The developed methods extend the current deep learning separation models and provide them with a better contextual perspective on musical data. Finally, the developed methods are evaluated on real-world music source separation data.

Additionally, the theoretical part of this thesis tries to provide a compact summery over the vast amount of literature existing in this field. Hopefully, this summary will help future students, who like to approach this field of research as well.

1.2. Human Perception of Auditory Scenes

The human brain is astonishingly good at separating and locating sound sources from a perceived sound mixture. On a crowded street there can easily be a hundred different sound sources. Driving cars, talking people, barking dogs, and many more are all superimposed and perceived as two pressure waves on our eardrums. Nevertheless the human brain can locate every sound source and focus on a single conversation inside this mixture without any problems. This task seems so natural to humans, that its complexity is often underestimated. Albert S. Bregman, the founder of the Auditory Scene Analysis Theory, puts this task into perspective with an example:

Imagine you stand at a lake without being able to see or hear anything, except the height of the arriving water waves. Then you are asked questions about what is happening on the lake. Are there people swimming in the lake and how many? Is the nearer one swimming from left to right or right to left? Did something fall into the water? You must answer these questions just from the movement of the arriving waves, without having any additional information. This would seem to be an impossible task. Yet consider a very similar problem. As you sit in a room your ear picks up the pressure waves from a lake of air surrounding you. Just like in the previous case, you are offered no information about the happenings except for the movements of these waves. Still, your brain makes it exceptionally easy to answer the same kinds of questions: Are there people talking in the room, and how many? Is the nearer one moving from left to right or right to left? (adapted from [1], page 34)

How can the human brain analyze and separate the surrounding sound mixture so well? Auditory Scene Analysis creates a theoretical foundation over the strategies being used by the human auditory system while organizing sound into perceptually meaningful elements. It describes the process with several grouping principles that decide if a sound is integrated into another element, or segregated, forming a new element. Auditory Scene Analysis is inspired by the principles of perceptual organization discovered by the school of Gestalt psychology.

After the sound mixture arrives on the eardrum, the first stage of analysis takes place in the cochlea, here the sounds are decomposed into separate neural patterns representing different frequencies [2]. This representation allows separating sound by its frequency content. For pure tones, e.g. sounds consisting of a single frequency, this process can already separate the mixture into sources. Although for almost all the sounds occurring in a natural environment this is not the case. They are constructed of complex spectra, containing lots of different frequency components. Figure 1.1 shows a piano note and a snare drum hit, as well as their resulting mixture, both in the time-domain representation (left side) and in the spectral representation (right side). Here it is easy to see that the sources can be separated much easier in the spectral domain. However, for complex mixtures containing many different sounds, this is still not a trivial task.

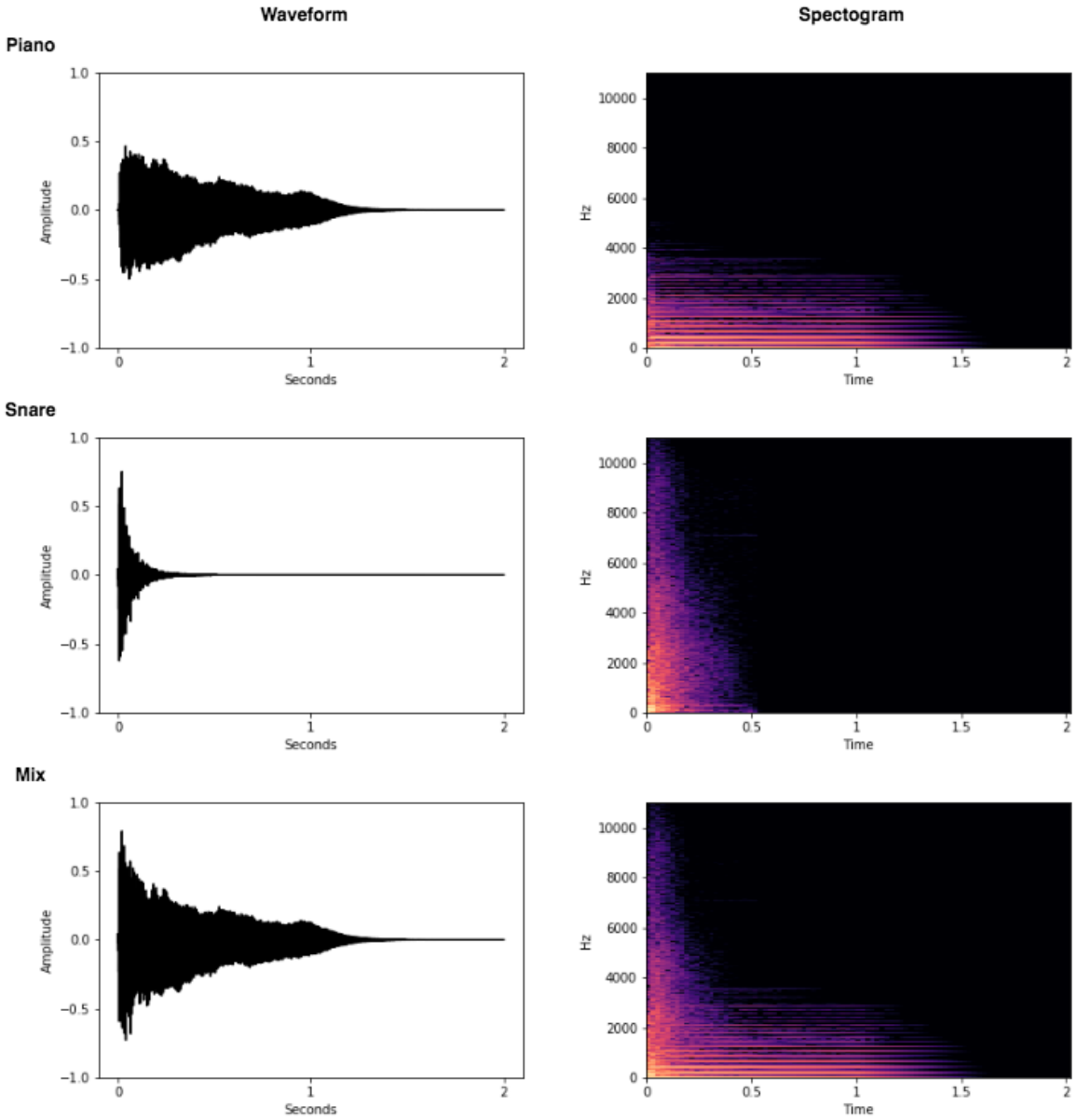

Figure 1.1.: Representation of a piano note, a snare hit, and their mixture. On the left side, the waveform in the time-domain and on the right side, the spectral representation in the frequency-domain are shown.

The core idea of Auditory Scene Analysis builds upon the grouping of different frequency parts to form sources. This perceptual organization happens both over time (sequential clustering) and over frequencies (simultaneous clustering). Hereby the distance in the spectral representation plays an important role. It shows that there is a tendency for similar sounds to group together where both nearness in frequency and in

time are reasons for this grouping process [1]. Although spatial distance is one obvious factor for building these perceptual clusters many other properties play a key role in this process, among them are:

1. Timbre (differences in the sound quality despite having the same pitch caused by differences in the harmonic series)
2. Spectral similarity (to what extent they share frequency components or similar frequency regions)
3. Temporal properties (such as the abruptness of onset of sounds)
4. Geometric location in space (the direction the sound is perceived from)
5. Intensity

Using these cues, human perception tends to create the 'simplest' clusters that it can, by grouping components together that show high similarities. Also, the history of already identified clusters plays an important role in the identification process. Only highly dissimilar sounds are explained by a new events. These clusters then form different auditory streams in our perception that one can focus on separately [1].

Bregman divides the cause for the formation of a cluster into three categories. The first two categories are based on sound recognition and can be seen as a recall of an already learned spectral pattern. He divides these into automatic recognition and voluntary recognition. Automatic recognition corresponds to the processes we subconsciously respond to, e.g. someone calling our name. Voluntary recognition on the other hand corresponds to the conscious process of focusing on one specific known sound, e.g. a conductor paying attention to a specific instrument in an orchestra. As it's not possible to store a blueprint for every sound and these blueprints have to be learned in the first place, there must be a more fundamental process underlying the auditory scene analysis. This is the third category that Bregman calls the primitive auditory scene analysis. It relies on general acoustic properties, which are given by physical laws. Our sensory system has evolved to take advantage of the regularities emerging from these rules [3].

In [2] Bergman points out four of these regularities which can be found in natural sounds, though he states that there still might be more unknown:

1. *Unrelated sounds seldom start or stop at exactly the same time.*
   (old-plus-new strategy)
2. *The gradualness of change:*
   - *A single sound tends to change its properties smoothly and slowly.*
   - *A sequence of sounds from the same source tends to change its properties slowly.*

3. *When a body vibrates with a repetitive period, its vibrations give rise to an acoustic pattern in which the frequency components are multiples of a common fundamental* (harmonic series).

4. *Many changes that take place in an acoustic event will affect all the components of the resulting sound in the same way and at the same time* (principle of common fate).

These regularities apply to the earlier mentioned properties which are measuring the similarity of different frequency components of a spectrum. Our brain tends to cluster sounds according to these rules and their similarity to create separate auditory streams. Bregman proves this with many different auditory experiments which can be found in [4] and also tried out interactively at `http://auditoryneuroscience.com/scene-analysis`.

1.3. Problem Formulation

Sound is a series of overlapping pressure waves propagated through the air (or any other medium). A microphone picks up these pressure differences over time and converts them into an electrical signal. This signal is measured at a fixed sampling frequency (for music typically at 44.1 kHz) and then recorded in a digital format. The time-series can be represented mathematically by a time-dependent variable: $x[t]$. If more than one microphone is used to record the sound pressure waves they can be written as a multi-channel signal in a vector notation: $\mathbf{x}[t] \in \mathbb{R}^I$, where I is the number of signals ($I = 2$ for stereo). During this thesis vectors, matrices and tensors will be written in **bold**. Due to the digital recording process, all signals will be written in a discrete form $x[t]$ and not as continuous signals $x(t)$.
In audio source separation, it is very common to use a Time-Frequency (TF) representation of the signal as sources tend to be less overlapped in the TF representation than in their waveform [5]. The Short-Time Fourier Transformation (STFT) is the most common TF representation. It transforms a fixed-length signal $x[t]$ into a complex matrix $\bar{\mathbf{X}} \in \mathbb{C}^{F \times N}$, with $F \times N$ so-called TF bins.[1] The angle of the complex-valued STFT accounts for the phase of the corresponding sinusoid at that TF bin, while the magnitude accounts for its amplitude in the signal. To index a TF representation either the bracket notation $\bar{\mathbf{X}}[f, n]$ or the subscript notation $\bar{x}_{f,n}$ will be used depending on the context. The latter subscript notation will also be used if there is more than one variable of the same nature. For example, the multi-channel signal $\mathbf{x}[t] \in \mathbb{R}^I$ can also be written with the subscript notation $x_i[t]$ with $i = 1, 2, ..., I$.

[1] Complex valued variables will always be written with a bar on top.

Auditory Scene Analysis shows that the separation process of sound mixtures into sources performed by the human brain relies on a lot of different strategies to validate itself. These strategies are based on regularities that are found in naturally occurring sounds and can be seen as a combination of physical laws and probabilistic assumptions about the world around them. These regularities create a shared evidence that is used to separate the mixture into perceptually meaningful elements. A simple experiment by Bregman shows that the auditory system can be easily fooled when sounds are not mixed according to these regularities:

"We start with a tone that glides up and down in frequency repeatedly. Then we remove a bit from each rising and each falling portion and replace it with a silent gap, causing discontinuities to be heard in the gliding tone "...". However, when loud noise bursts are inserted where the gaps were "..." the tone is heard as complete, gliding right through them."..." The presence of an 'occluder' – in this case a sound that might have obliterated parts of the tone – is interpreted as hiding the tone, and the brain restores what it predicts to be missing." (in [1], page 34)

This proves how much the auditory system relies on these assumptions rather than having a fixed solution to the source separation problem. A very similar observation can be found when trying to solve the Audio Source Separation problem mathematically. This field of research is called Blind Source Separation and is often described with the cocktail party problem. Imagine there are J persons at a cocktail party, each of them is talking, representing a separate sound source. The individual sound sources can be seen as signals and written down in a vector notation $\mathbf{s}[t] = [s_1[t], s_2[t], ..., s_J[t]]^T$. Inside the room are I microphones capturing the auditory scene through the observed signals $\mathbf{x}[t] = [x_1[t], x_2[t], ..., x_I[t]]^T$. Each observed signal captures a mixture of all the source signals. In a very naive way this process can be written down as:

$$x_i[t] = \sum_{j=1}^{J} a_{i,j} s_j[t] \qquad \text{with } i = 1, ..., I \tag{1.1}$$

And in a more compact vector-matrix notation, with mixing matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$, it can be written as:

$$\mathbf{x}[t] = \mathbf{A}\mathbf{s}[t] \tag{1.2}$$

Equation 1.1 and 1.2 describe the stationary instantaneous mixture process. This means that all observed signals (e.g. the microphone signals) at time t are linear combinations of the source signals (e.g. the speakers) at time t. Therefore the observed signals only differ in the loudness of their contained sources described by mixing coefficients $a_{i,j}$. These differences can be explained by the distances between speakers and microphones, where $a_{i,j}$ describes the amount of signal of speaker j contained on microphone i.

The stationary instantaneous mixture process is inaccurate for almost all-real world scenarios. This is mainly due to the laws of acoustics. As sound travels at around 343 meters per second, different delays occur on the observed signals. Also, any real environment, such as a room, has physical boundaries, e.g. walls, floor, ceiling. The sound waves emitted by a source are reflected on these boundaries and form new sources, which are delayed and filtered versions of the original source signal. This process is called reverberation. These two factors can be included in the mixture process by using a convolutive model:

$$x_i[t] = \sum_{j=1}^{J} \sum_{k=1}^{L} a_{i,j}[k] s_j[t-k] \qquad \text{with } i = 1, ..., I \tag{1.3}$$

And written in a more compact version using the convolution operator $*$ and Finite Impulse Response (FIR) filter coefficient vector $\mathbf{a}_{i,j} \in \mathbb{R}^L$:

$$x_i[t] = \sum_{j=1}^{J} (\mathbf{a}_{i,j} * s_j)[t] \qquad \text{with } i = 1, ..., I \tag{1.4}$$

Here L defines the length of the FIR filter $\mathbf{a}_{i,j}$ and should be as long as the last reflection that occurs inside the acoustical environment. It is identical to the impulse response of the acoustical environment taken on the position of the speaker j with microphone i. The convolutive mixing model is accurate for a stationary scene, although it cannot model moving sources. If sources are moving inside the room, a non-stationary model has to be used by making FIR filter $\mathbf{a}_{i,j}$ time depended:

$$x_i[t] = \sum_{j=1}^{J} (\mathbf{a}_{i,j,t} * s_j)[t] \qquad \text{with } i = 1, ..., I \tag{1.5}$$

To account for noise in the recording process an additive noise term can be added to the mixing model as well:

$$x_i[t] = \sum_{j=1}^{J} (\mathbf{a}_{i,j,t} * s_j)[t] + \eta \qquad \text{with } i = 1, ..., I \tag{1.6}$$

Equation 1.6 is a quite realistic model of a natural mixing process, accounting for the slow propagation of sound, delays, reverberation, and noise. A major problem with this model is the massive amount of parameters it contains. There are $I \times J \times L \times T$ FIR coefficients, where T is the length of the audio recording in samples. Furthermore, the additive noise term makes it very difficult to solve the equation for the source signals $s_j[t]$ [6]. As a consequence for most of the Blind Source Separation solution, either the stationary instantaneous mixture model 1.2 or the stationary convolutive mixture model 1.4 with small FIR coefficient lengths L are used.

A special case arises in the case of professionally mixed music. Here it is very common to 'master' the final mixdown, which involves applying different signal processing steps to ensure 'better' sound quality. These processing steps include compression of the mixture waveform, as well as other non-linear processing methods. In these mixtures, a non-linear function has to be used to account for these processing steps:

$$x_i[t] = f(\sum_{j=1}^{J} a_{i,j} s_n[t]) \qquad \text{with } i = 1, ..., I \tag{1.7}$$

Equation 1.1 - 1.7 are all different approximations to model the mixing process of the observed signals $x_i[t]$. Generally, the Blind Source Separation problem is defined as the procedure of estimating the original signals $\mathbf{s}[t]$ through the observed signals $\mathbf{x}[t]$ [6]. Referring back to the earlier stated cocktail party problem: one wants to restore the original conversations from the observed mixture signals that are recorded on the microphones.

Using model 1.2 and in the case where there are as many sources as there are microphones $J = I$, it is easy to see that one simply needs to invert mixing matrix $\mathbf{A}$ to find the sources:

$$\mathbf{A}^{-1}\mathbf{x}[t] = \mathbf{s}[t] \tag{1.8}$$

However, one seldom has the correct information over the mixing processes, so the mixing matrix $\mathbf{A}$ is unknown. The source signals $\mathbf{s}[t]$ are unknown as well, as in the case of known sources the problem formulation would be unnecessary. Therefore, the problem is mathematically ill-posed as there are two unknowns variables and only one known variable [7]. Hence, an infinite set of solutions (pairs of source signals and mixing matrix) exists which create the observed mixture signals. This conclusion also holds for all other models in equation 1.3 - 1.7 [8].

Following this conclusion, it can be seen why the human auditory system has to rely on assumptions about the source signals and the mixing process, as without them the problem cannot be solved. This breaks down the audio source separation problem into a search for valid assumptions over the source signals, and the mixing process which results in a unique solution to the blind source separation problem.

2. Theoretical Background

There are many approaches to the source separation problem since this field of research has enjoyed tremendous research activity for roughly 50 years. Because this Thesis deals with the specific problem of music source separation, only the main approaches related to this problem will be mentioned here.

In general, solutions to the audio source separation problem can be divided into model-based and data-driven approaches. The former tries to capture the source signals and the mixing process with a particular mathematical model. The observed signals are then used to derive the best fitting model parameters, for example in a Maximum Likelihood (ML) manner. On the other hand, the data-driven approach exploits large databases of audio examples where the isolated source signals are available. This enables the use of machine learning methods to learn how to separate. These two different approaches also reflect Bregman's division into primitive and recognition based auditory scene analysis. This chapter starts with a historical overview of model-based approaches to the audio source separation problem in 2.1 and then moves on to modern deep learning-based solutions in 2.2, which are achieving the current State-of-the-Art results.

2.1. Early Approaches

Introduction to Audio Source Separation

The two main goals that are driving the development of new solutions to the audio source separation problem are Speech Separation and Music Separation. Respectively there are two evaluation campaigns, the CHiME Speech Separation Competition[1], and the SigSep Music Source Separation Competition[2]. Because of its application as a preprocessing step in natural language processing, speech separation has got more attention over the last decades, however, most of the separation models can be applied to both problems or can be easily adapted by changing the underlying mixing model, the number of observed signals and the number of source signals [9].
The separation models mostly vary in their fundamental assumptions made to solve the blind source separation problem (see chapter 1.3). While computational auditory scene analysis tries to specifically replicate the functionality of the human auditory system [10], most models are based inside a more general framework and include insights of auditory scene analysis through specific mathematical constraints.

[1] `https://chimechallenge.github.io/chime6/`
[2] `https://sigsep.github.io/`

To show how these constraints are developed and to give a short example of how a solution to the audio source separation problem might look let's consider the simplified problem of speech enhancement and its counterpart in music the vocal/accompaniment separation. In this problem, the aim is to recover a voice signal which got corrupted by noise. For simplicity let's also assume the instant mixture model given in equation 1.1. Given two sources, e.g. voice and noise, and only one microphone, we can simplify the model even more by already including the mixing coefficients into the source signals:

$$x[t] = s[t] + n[t] \tag{2.1}$$

Here $s[t]$ is the voice signal, which we want to recover, while $n[t]$ contains the residual noise or the musical accompaniment. A very common method is to take the STFT of the mixture signal resulting in the complex-valued matrix $\bar{\mathbf{X}} \in \mathbb{C}^{F \times N}$ and perform the separation only on the magnitude information $|\bar{\mathbf{X}}| \in \mathbb{R}^{F \times N}$ [1]. Afterwards, the phase information of the mixture signal is used to perform the inverse STFT. Furthermore, it is often assumed that the magnitude spectra of the sources sum up to the mixture magnitude spectrum, which is wrong as this ignores interference effects caused by different phases of the source signals. Moreover, one can now assume that the noise signal is stationary and only consisting of a few non-varying elements (low rank). For background noise in the speech enhancement problem, this might be a valid assumption. On the other hand the accompaniment signal of normal music is definitely not stationary, but it often has a very repetitive nature, so the low-rank assumption still holds. The speech signal, on the contrary, has more variations (higher rank) but is relatively sparse in the TF representation. The Robust Principle Component Analysis (RPCA) model captures both these assumptions [11]. It models the low-rank noise magnitude matrix $\mathbf{N} \in \mathbb{R}^{F \times N}$ with the nuclear norm (sum of singular values) and the sparse speech magnitude matrix $\mathbf{S} \in \mathbb{R}^{F \times N}$ with the L1-norm (sum of absolute values of matrix entries) [12]. This results in the following optimization problem:

$$\begin{aligned} \text{minimize:} \quad & ||\mathbf{N}||_* + \lambda ||\mathbf{S}||_1 \\ \text{subject to:} \quad & |\bar{\mathbf{X}}| = \mathbf{S} + \mathbf{N} \end{aligned} \tag{2.2}$$

Solving this optimization problem to obtain $\mathbf{S}$ and $\mathbf{N}$ has shown to separate a voice signal from background noise in cases where the source signals follow these assumptions. Although a lot of partly erroneous simplifications have been made to the original mixture model, this model can be used to separate the lead vocal from the rest of the musical accompaniment, as shown in [12], or to separate a speech from a noisy background, as shown in [13].

Separation models can be roughly divided into multi-channel-filtering, which mostly relies on spatial information for the separation, and time-frequency-filtering, which mostly relies on spectral information for the separation (like the RPCA) [14]. In the next two subsections, the most prominent multi-channel- as well as time-frequency-filtering models are summarized.

[1] F being the number of frequency bins in the STFT and N being the number of STFT windows

Independent Component Analysis

The Independent Component Analysis (ICA) is a very generalized statistical technique to extract independent source factors from a mixture and is closely related to Projection Pursuit. It is one of the first and also most general solutions to the blind source separation problem and a variety of ICA models exist. Because the ICA relies on a very weak statistical assumption about the sources, it is outperformed by most other separation models in the case of audio source separation. This section will just give a brief historical overview of the basics of the ICA.

Hyvarinen et al. showed that it is sufficient to assume that the sources $\mathbf{s}_j[t]$, at each time instant t, are statistically independent [15]. In the ICA model, the source signals and the observed signals are treated as random variables, discarding the time-series information and treating every time point as a realization of the random variable. This makes the model also applicable to data without a time structure. The sources are statistically independent if the realization of one does not affect the probability distribution of the other. This can be stated in terms of their joint probability distribution as follows:

$$p(s_1, s_2, ..., s_J) \;=\; p(s_1)p(s_2)...p(s_J) \tag{2.3}$$

In general, the ICA problem is defined by solving for the unmixing matrix $\mathbf{W} = \mathbf{A}^{-1}$ from equation 1.8, which will yield the independent components of the mixture. This results in the following solution:

$$\mathbf{W}\mathbf{x} = \mathbf{s} \tag{2.4}$$

There are a couple of limitations arising through this solution:

1. The number of observed signals I has to be bigger or equal to the number of source signals J (so that $\mathbf{A}$ is invertible). Although there are approximations to the so-called ICA with an overcomplete basis (see [8] - chapter 16).

2. The order of the independent components cannot be determined. This is a simple result from the fact that both $\mathbf{s}$ and $\mathbf{W}$ are unknown.

3. The variances (energies) of the independent components cannot be determined (due to the same reasons as stated in 2.). Instead, the variance of the sources is fixed $\mathbb{E}\{\mathbf{s}^2\} = 1$.

4. The sign of the independent components cannot be determined (due to the same reasons as stated in 2.).

The ICA starts by whitening the data (zero mean and identity covariance matrix). This converts the unmixing matrix $\mathbf{W}$ into an orthogonal matrix and consequently cuts the number of estimated parameters down from J^2 to $J(J-1)/2$ [8].

An approach to solve for the independent components is derived from the Central Limit Theorem, which can be summarized as follows: The sum of two independent random variables tend to have a distribution that is closer to a Gaussian distribution than the separate original variables [16]. Hence one needs to find a projection that maximizes the non-gaussianity of $\mathbf{w}_j^T\mathbf{x}$ ($\mathbf{w}_j$ being a row of the unmixing matrix $\mathbf{W}$), which in case of statistical independence equal one of the sources [15].

One way to measure the non-gaussianity of the projection is the kurtosis. The kurtosis is closely related to the fourth moment of a probability distribution and measures the tails of the distribution (high values in kurtosis correspond to large amounts of outliers - i.e. data points far away from the mean). In general, Gaussians are almost the only distributions which have a kurtosis of zero [15]. In practice, one could start with a random vector $\mathbf{w}_j$ and then optimize in the direction of the steepest growth of the absolute value of the kurtosis with a gradient method. The independent components can be either found one at a time or all together by decorrelating the estimated unmixing matrix $\mathbf{W}$ after every gradient step, for example with Gram-Schmidt orthonormalization [8]. Because the kurtosis relates to the tails of the distribution, its value only relies on a few data points furthest away from the mean. This makes the kurtosis not very robust and very unstable.

A more stable way to measure the non-gaussianity of random variable involves using the differential entropy of its probability distribution. The entropy of a random variable is related to the information that the observation of the variable gives. The more unpredictable and unstructured the variable is, the larger its entropy. Gaussian variables have the largest entropy among all random variables of equal variance [8]. This can be used to set up a new measure for non-gaussianity using the difference in differential entropy of the projection $\mathbf{w}_j^T\mathbf{x}$ and a Gaussian variable of the same variance. This measure is called Negentropy. It is always non-negative and only zero when the observed variable is Gaussian. Negentropy can be seen as an optimal measure for non-gaussianty [8]. The problem in using Negentropy is, that the probability density function underlying the data needs to be known, which is rarely the case. Therefore, it needs to be approximated. Hyvärinen has shown, that it can be approximated by several non-linear functions, which can be chosen to have good statistical and computationally properties [17]. This approximation leads to the FastICA algorithm, which is a popular and efficient solution to the ICA problem [15].

Over the last decades, a lot of alternative approaches to the ICA problem were discovered including minimization of mutual information, maximum likelihood estimation and natural gradient methods, nonlinear decorrelation, separation by autocovariance (using time structure and autocorrelation), and tensorial eigenvalue decompositions [8]. They all show some equivalences in their solutions despite the difference in their approaches and will be not discussed furthermore in this thesis.

For audio source separation often the convelutive mixture model is required, as described in equation 1.3. In the convolutive case the ICA solution from equation 2.4 does not hold anymore, because a deconvolutive FIR Filter is needed for every source signal. A simple way to solve this problem can be achieved by using a fixed length FIR filter of L samples and stacking up the delayed versions of the signal in the following way:

$$\begin{aligned}\widetilde{\mathbf{x}}[t] \quad = \quad & [\ x_1[t], x_1[t-1], ..., x_1[t-L+1], x_2[t], x_2[t-1], ..., x_2[t-L+1], \\ & ..., x_I[t], x_I[t-1], ..., x_I[t-L+1]\ ]^T\end{aligned} \tag{2.5}$$

$$\begin{aligned}\widetilde{\mathbf{s}}[t] \quad = \quad & [\ s_1[t], s_1[t-1], ..., s_1[t-L+1], s_2[t], s_2[t-1], ..., s_2[t-L+1], \\ & ..., s_J[t], s_J[t-1], ..., s_J[t-L+1]\ ]^T\end{aligned} \tag{2.6}$$

Now the ordinary instantaneous ICA model can be applied to the stacked vectors with the new mixing matrix $\widetilde{\mathbf{A}} \in \mathbb{C}^{IL \times JL}$:

$$\widetilde{\mathbf{x}} = \widetilde{\mathbf{A}}\ \widetilde{\mathbf{s}} \tag{2.7}$$

In theory, solving this ICA problem should simultaneously separate the sources and perform a deconvolution. However, in practical applications, L needs to be quite large (ten thousands of samples in length) to accurately model the reverberation of an acoustical space. This especially affects the dimensions of mixing matrix $\widetilde{\mathbf{A}}$, increasing the computational complexity, as the number of parameters to be estimated grows with $I \times J \times L^2$. This problem can not be avoided, as it's a result of the convolutive mixing model [8].
Another approach to solve the convolutive ICA problem is to take the Fourier transform, as convolutions are products between Fourier transforms in the frequency domain [18]. In practice the Short Term Fourier Transform (STFT) is used, resulting in the new complex-valued ICA problem:

$$\bar{\mathbf{x}}[f, n] = \bar{\mathbf{A}}[f]\ \bar{\mathbf{s}}[f, n] \tag{2.8}$$

To correctly represent the convolutive mixture process, the STFT window size N has to be bigger than the length of the FIR mixing filters L. This concept is called the narrowband approximation and it gets more accurate when the sizes of the mixing filters impulse responses are small $N >> L$ [19]. Otherwise, if $N < L$ the separation performance degrades [20].

There are a few consequences arising through the complex ICA model. Firstly, and fortunately, the distributions of the transformed audio signals in the frequency domain are by far more non-Gaussian than the distribution of the time domain signal [6]. This increases the performance of the ICA model and improves convergence. Secondly, and more problematic, the before mentioned permutation and scaling invariance, which is not a big problem in the time domain, has now severe consequences. As mixing matrix

$\bar{\mathbf{A}}[f]$ becomes a function of the frequency, it needs to be solved for every frequency bin $f = 1, 2, ..., F$ separately. The permutations (i.e. the ordering of the estimated sources) and signs of the sources are usually different in each frequency bin solution of $\bar{\mathbf{A}}[f]$. For the reconstruction of a source signal $s_j[t]$ in the time domain, one needs all its frequency components. Hence, a method to align the permuted sources in different frequency bin solutions is needed [8]. Different methods were proposed, most of them providing a method for grouping frequency bins by their overall energy envelope [6].

While ICA is used in biomedical signal processing, image processing, telecommunications, econometric, data analysis, and feature extraction, its performance in audio source separation is in general not satisfying [8]. This is due to various issues with the audio mixing model as discussed in 1.3:

1. The convolutive mixing model complicates the ICA solution.
2. The additional sources caused by the reflections are not independent anymore.
3. Typical recordings are made with two microphones only, while the amount of sources contained on the recording is much larger.
4. For most cases, a non-stationary mixing model (equation 1.5) needs to be used, as sources move in space.
5. In the case of a low Signal to Noise Ratios, the noisy mixing model is required (equation 1.6).
6. In the case of music, the non-linear mixing model (equation 1.7) is needed, complicating the ICA solution even more (see [8] - chapter 17).

Due to these complications, it may be that the assumption of independence of the source signals, is not enough for sound separation. The convolutive mixing model with a large number of free parameters and a dynamically changing mixing matrix requires more information on the signals and the mixing matrix [8]. This can be also seen in [21], where musical sources can be separated quite well on artificially made instantaneous mixtures by ICA, but as soon as the model is applied to real-world conditions (e.g. propagation delays and reverberation) it fails.

Non-Negative Matrix Factorization

Since the ICA model is not well suited for the problems arising through real-world audio signal mixtures other models were developed in order to take these issues into account. Especially the Non-Negative Matrix Factorization (NMF) framework has shown good capabilities to solve the audio source separation problem under different circumstances with good performance [9]. In contrast to the ICA, NMF was not specifically developed to solve the blind source separation problem, instead, it is a linear dimensionality reduction technique for non-negative data. It has become popular because of its ability to automatically extract sparse and easily interpretable factors out of the data [22]. Particularly through the multiplicative update rules proposed by Lee and Seung [23], which guarantee a fast convergence, NMF became a widely used technique for data analysis and has been used in many fields. Data compression models like NMF logically should be suitable for source separation, since in both cases the models tend to reduce the redundancy in the signal [19]. The early work of Smaragdis and Brown adapted the NMF model to the field of Music Information Retrieval (MIR) [24] and initiated the diverse use of NMF-based research, especially in audio source separation.
The fundamental constraint of NMF, as already suggested by the name, is the non-negativity of the data matrix $\mathbf{X} \in \mathbb{R}_{\geq 0}^{F \times N}$, where in general F is the dimensionality of the data and N is the number of data points. The data matrix $\mathbf{X}$ is approximated by two smaller matrices:

$$\mathbf{X} \approx \hat{\mathbf{V}} = \mathbf{WH} \tag{2.9}$$

Where $\mathbf{W} \in \mathbb{R}_{\geq 0}^{F \times K}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{K \times N}$ are non-negative and of a lower rank than $\mathbf{X}$. The Hyperparameter K is usually chosen so that $FK + KN << FN$, yielding a low-rank approximation $\hat{\mathbf{V}}$, that can only be achieved if the basis vectors $\mathbf{W}$ discover hidden structures inside the data [23]. $\mathbf{W}$ contains K basis vectors in its columns that represent these structures. They can be seen as elementary building blocks of the data. $\mathbf{H}$ on the other hand contains the weights of these basis vectors for every sample $n = 1, 2, ..., N$. To avoid scaling indeterminacy between different NMF components the columns of $\mathbf{W}$ are often normalized to have a magnitude of 1.
In audio source separation $\mathbf{X}$ typically represents the magnitude or power spectrogram of the audio mixture with F being the frequency axis and N the time axis of the STFT. In the single-channel case ($I = 1$) the NMF factorizes the spectrogram $\mathbf{X}$ into K prototypical spectral patterns stored in the columns of $\mathbf{W}$ and their activation's over time stored in the rows of $\mathbf{H}$. The prototypical spectral patterns tend to represent the sources of the mixture. This decomposition into spectral building blocks and temporal activation functions can be seen in figure 2.1, where a short piano sequence is separated into its four root notes as well as a noisy attack sound.

In general, NMF is ill-posed and NP-hard. Hence there is no closed-form solution and non-linear optimization methods are used to solve the factorization problem in an iterative way which might only converge to stationary points (e.g. local minima) [22].

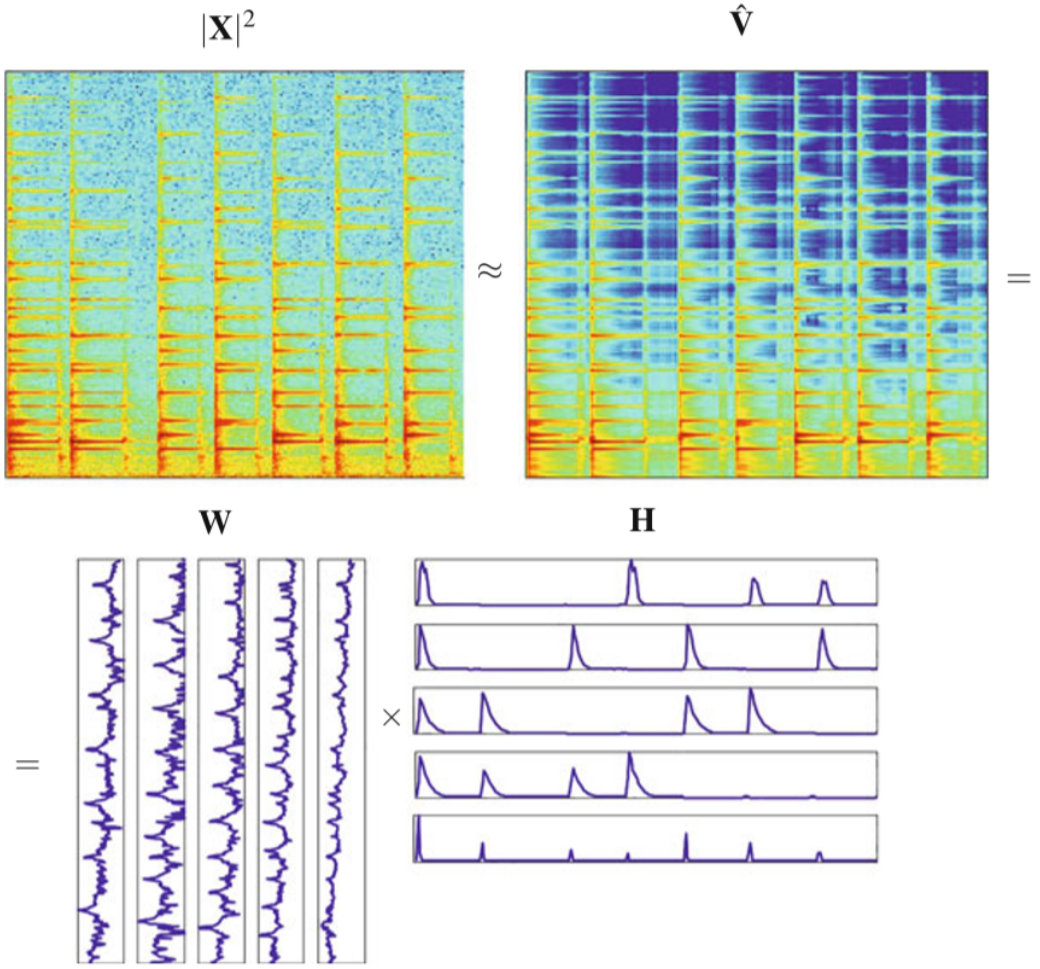

Figure 2.1.: NMF applied to the spectrogram of a short piano sequence composed of four notes (Taken from [25]).

The NMF problem can be stated in a minimization problem:

$$\min_{\mathbf{W},\mathbf{H}} D(\mathbf{X}|\hat{\mathbf{V}}) \text{ subject to } \hat{\mathbf{V}} = \mathbf{WH} \text{ and } \mathbf{W}, \mathbf{H} \geq 0 \tag{2.10}$$

For the optimization process a cost function $D(\mathbf{X}|)$ is needed to measure the closeness of approximation $\hat{\mathbf{V}}$ to the real data $\mathbf{X}$. The most trivial choice is the Euclidean Distance which provides convenient mathematical properties:

$$D(\mathbf{X}|) = \sum_{f=1}^{F} \sum_{n=1}^{N} (\mathbf{X}[f,n] - \hat{\mathbf{V}}[f,n])^2 \tag{2.11}$$

Another very commonly used cost function is motivated by the probabilistic Kullback-Leibler (KL) divergences and given by:

$$D(\mathbf{X}|\hat{\mathbf{V}}) = \sum_{f=1}^{F}\sum_{n=1}^{N}(\mathbf{X}[f,n]\log\frac{\mathbf{X}[f,n]}{\hat{\mathbf{V}}[f,n]} - \mathbf{X}[f,n] + \hat{\mathbf{V}}[f,n]) \tag{2.12}$$

In audio source separation the Itakura-Saito (IS) divergence has shown good results as it is scale-invariant, i.e. $D(\lambda\mathbf{X}|\lambda\hat{\mathbf{V}}) = D(\mathbf{X}|\hat{\mathbf{V}})$ [26]. In contrast to the former ones it takes both low- and high-power components inside the data to the same amount into account which is beneficial given the logarithmic scaling of loudness. The IS divergence is given by:

$$D(\mathbf{X}|\hat{\mathbf{V}}) = \sum_{f=1}^{F}\sum_{n=1}^{N}(\frac{\mathbf{X}[f,n]}{\hat{\mathbf{V}}[f,n]} - \log\frac{\mathbf{X}[f,n]}{\hat{\mathbf{V}}[f,n]} - 1) \tag{2.13}$$

All three former cost functions can be generalized by a continuous family of functions: the β-divergence [25]. To solve the optimization problem in equation 2.10 one can simply use gradient descent to minimize the respective cost function. The famous multiplicative update rule from Lee and Seung arises through gradient descent with an adaptive re-scaled learning rate and "*can also be interpreted as a multiplicative rule with the positive component of the gradient in the denominator and negative component as the numerator of the multiplicative factor*" (in [23], page 3). The multiplicative updates ensure the non-negativity of $\mathbf{W}$ and $\mathbf{H}$, yield an optimal convergence speed, and do not require a manually set learning rate. The update rules for the specific cost functions and their proofs of convergence are shown in [23] and for the IS-divergence in [26].

After the iterative estimation of the NMF components the sources are separated from the mixture in the minimum mean square error sense via Wiener filtering [7]:

$$s_{j,f,n} = \frac{w_{j,f}h_{j,n}}{\sum_{i=1}^{J} w_{i,f}h_{i,n}}\bar{x}_{f,n} \tag{2.14}$$

Where $w_{j,f}$ is a coefficient of $\mathbf{W}$ representing the spectrum of source j and $h_{j,n}$ is the corresponding coefficient of $\mathbf{H}$ representing the intensity of that source at time frame n. $\bar{x}_{f,n}$ is the complex-valued STFT value of the audio mixture at frequency bin f and time frame n. In this case, the number of sources equals the number of NMF components ($K = J$), though in most cases the choice of the hyperparameter K is not so trivial [22]. In theory each column $\mathbf{w}_k$ with $k = 1, 2, ..., K$ reflects an elementary building block of the data. For audio signals, however, each sound source rarely consists of a single NMF component (i.e. a single spectrum). For example, a musical instrument typically involves several notes with different pitches, while a speech source involves several phonemes. Therefore individual NMF components need to be clustered together into their respective sources [25].

This clustering problem can be avoided by using a fixed pre-defined source model. The so-called supervised NMF makes use of source-specific dictionaries, which contain prototypical spectra of the sources. By keeping the pre-defined dictionary $\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_J]$ fixed during estimation the optimization problem becomes convex and can be solved much faster [22]. The source-specific dictionaries must be learned in advance. This requires isolated audio recordings of all the sources. Usually, the dictionaries are learned by performing a prior NMF over the individual sources. After this NMF factorization $\mathbf{W}$ is kept for each source, while $\mathbf{H}$ is discarded. There are other methods to create the source-specific dictionaries, which can be found in [27]. The supervised NMF model suffers from one major limitation: unless regularisation such as sparsity is enforced the number of dictionary elements must be smaller than the number of frequency bins [25]. If the number is bigger the dictionary might span up an orthogonal basis and does not utilize any hidden structures in the data anymore. To fully resemble all characteristics of an instrument or speech, a quite large number of prototypical spectra is needed. The supervised NMF does not generalize well, as it can only represent sources by a linear combination of its dictionary elements.

Another way to avoid the problem of grouping the NMF components $\{\mathbf{W}_j, \mathbf{H}_j\} \subsetneqq \{\mathbf{W}, \mathbf{H}\}$ to the correct sources j is to include the clustering process directly into the optimization objective. It is assumed that instead of representing each source with individual NMF components $\{\mathbf{W}_j, \mathbf{H}_j\}$ all the sources share the same NMF components $\{\mathbf{W}, \mathbf{H}\}$ [28]. To determine the relationship of NMF component k to source j a third matrix $\mathbf{Q} \in \mathbb{R}_{[0,1]}^{K \times J}$ is included in the factorization:

$$\mathbf{V} = \sum_{k=1}^{K} \mathbf{w}_k \circ \mathbf{h}_k \circ \mathbf{q}_k \tag{2.15}$$

Where $\circ$ denotes the outer tensor product and $\mathbf{V} \in \mathbb{R}_{\geq} 0^{F \times N \times J}$ is defined as a three dimensional tensor in this case. The columns of $\mathbf{Q}$ are normalized ($\sum_{j=1}^{J} \mathbf{q}[j]_k = 1$). This ensures that mixture $\mathbf{X}$ is obtained by summing over the source dimension of $\mathbf{V}$. This factorization model is sometimes referred to as the Non-negative Tensor Factorization (NTF) [28]. It has the advantage that one does not need to specify the amount of NMF components K_j per source anymore, as the model will allocate them jointly with the optimization. Though for a correct NMF component allocation $\mathbf{Q}$ needs to be sparse (i.e., only a few non-zero components in the corresponding column). Thus, a sparsity-inducing penalty on $\mathbf{Q}$ to the corresponding optimization criterion should be added [19]. There are a lot of different NMF/NTF models for audio source separation and its equivalent probabilistic formulation the Probabilistic Latent Component Analysis (PLCA). For the length of this thesis only the multi-channel NMF from Ozerov, Vincent, and Bimbot presented in [7] and [29] will be discussed here as they generalizes several different audio source separation models and are also the basis of the audio source separation toolbox *FASST*[1] for MATLAB and Python.

[1] `http://bass-db.gforge.inria.fr/fasst/`

Their proposed model is based on a different mixing model than the ones presented in chapter 1.3. Instead of a single source signal, the mixing model is built on spatial source images:

$$\mathbf{x}[t] = \sum_{j=1}^{J} \mathbf{s}_j[t] \tag{2.16}$$

With spatial source image vectors $\mathbf{s}_j \in \mathbb{R}^I$ for every source $j = 1, 2..., J$. In contrast to the former mixing models which only account for point sources, this accounts for diffuse sources as well as sources with directivity patterns that are not spherical [7]. The before mentioned single channel NMF (equation 2.9) only relies on the non-negative magnitude or power spectrograms while completely discarding the phase information of the STFT. This inherently assumes that the source STFTs have the same phase or that only one component is active in every time-frequency tile [30] [31]. In the multi-channel NMF, the source spectrograms are modeled with the non-negative model while the mixing system is modeled with a complex-valued system. Every complex-valued STFT coefficients $\mathbf{s}_{j,f,n} \in \mathbb{C}^I$ is treated as realizations of zero-mean circular complex-valued Gaussian random variable with a structured variance (via NMF/NTF) and covariance [19]:

$$\mathbf{s}_{j,f,n} \sim \mathcal{N}(0, \bar{\mathbf{R}}_{j,f,n} v_{j,f,n}) \tag{2.17}$$

This is called the Local Gaussian Model (LGM) in which the matrix $\bar{\mathbf{R}}_{j,f,n} \in \mathbb{C}^{I \times I}$, called the spatial covariance matrix, represents the spatial characteristics of the source, and the non-negative scalar $v_{j,f,n} \in \mathbb{R}_{\geq 0}$, called spectral power, represents the spectral characteristics of the source [7]. The source STFT coefficients $\mathbf{s}_{j,f,n}$ are assumed to be mutually independent (conditionally to their covariance structure) over time, frequency, and between sources [29].

The spatial covariance matrix $\bar{\mathbf{R}}_j$ can be adapted to different mixing models. In the simplest case, it can be modeled with a rank-1 matrix: $\bar{\mathbf{R}}_j = \mathbf{a}_j \mathbf{a}_j^T$, where $\mathbf{a}_j$ is a column vector of mixing matrix $\mathbf{A}$ from equation 1.1. This model only accounts for point sources in an instantaneous mixing setup. To account for convolutive mixtures one can create a spatial covariance matrix for every frequency bin $\bar{\mathbf{R}}_{j,f}$. This model follows the narrowband approximation and is only valid if the length of the convolutive FIR filter L is smaller than the STFT window length. In contrast, if $\bar{\mathbf{R}}_{j,f}$ is modeled by a full rank matrix, the model also takes long reverberations and diffuse sources into account [7]. Similar to the ICA model, the spatial covariance matrix can also account for non-stationary mixing setups through a time-depended covariance matrix $\bar{\mathbf{R}}_{j,f,n}$, again, this will increase the number of free model parameters dramatically. In contrast to the ICA model the multi-channel NMF model can easily handle a variety of mixing situations: single-channel ($I = 1$), underdetermined ($1 < I < J$), and (over-)determined ($I \geq J$) [29].

In order to account for different source types several models have been proposed to model the spectral power structure $v_{j,f,n}$, including: unconstrained models, block constant models, Gaussian mixture models or hidden Markov models, harmonic NMF/NTF, temporal activation constrained NMF/NTF and source-filter models [29]. The multi-channel NMF model in [7] generalizes most of them by factorizing the spectral power into eight non-negative matrices. First, the spectral power is decomposed into an excitation spectral power $v^{excit}_{j,f,n}$, and a filter spectral power $v^{filt}_{j,f,n}$:
[19]:

$$v_{j,f,n} = v^{excit}_{j,f,n} \cdot v^{filt}_{j,f,n} \tag{2.18}$$

This reflects the source-filter model where for example the voice is modeled by an excitation source, e.g. the glottal source, filtered by a transfer function, e.g. the vocal tract. Each of them is structured by the NMF model as K characteristic spectral pattern $w_{j,f,k}$ modulated in time by weights $h_{j,k,n}$ [19]:

$$v_{j,f,n} = \sum_{k=1}^{K} w_{j,f,k} \cdot h_{j,k,n} \tag{2.19}$$

To further constrain and guide the factorization, each of the factors is modeled by two more sub-factors. The characteristic spectral pattern $w_{j,f,k}$ is further factorised into B elementary narrowband spectral patterns $e_{j,f,b}$ and their respective weights $u_{j,b,k}$. These narrowband patterns may be for instance harmonic, inharmonic, or noise-like components with a smooth spectral envelope [7]. The time varying weights $h_{j,k,n}$ are also further factorized into M time localized patterns $p_{j,k,m}$, and their respective time-varying weights $g_{j,m,n}$. These can be typical exponential and linear decay patterns to ensure temporal continuity. Alltogether the whole factorization of $v_{j,f,n}$ can be written as follows [19]:

$$v_{j,f,n} = \sum_{k=1}^{K} (\underbrace{(\sum_{b=1}^{B} e_{j,f,b} u_{j,b,k})}_{w_{j,f,k}} \cdot \underbrace{(\sum_{m=1}^{M} p_{j,k,m} g_{j,m,n})}_{h_{j,k,n}}) \tag{2.20}$$

This sub-factorization applies to both the excitation spectral power $v^{excit}_{j,f,n}$ and the spectral filter power $v^{filt}_{j,f,n}$. All sub-factors $e_{j,f,b}$, $u_{j,b,k}$, $p_{j,k,m}$ and $g_{j,m,n}$ for the excitation and filter spectral power respectively can be rewritten in matrix form to construct the final spectral power factorization model:

$$\mathbf{V}_j = \mathbf{V}^{excit}_j \odot \mathbf{V}^{filt}_j = (\mathbf{E}^{excit}_j \mathbf{U}^{excit}_j \mathbf{P}^{excit}_j \mathbf{G}^{excit}_j) \odot (\mathbf{E}^{filt}_j \mathbf{U}^{filt}_j \mathbf{P}^{filt}_j \mathbf{G}^{filt}_j) \tag{2.21}$$

Here, $\odot$ denotes the elementwise matrix multiplication (Hadamard product). The multi-channel NMF model can be easily adapted to different modeling situations. This can be done by using different constraints for the sub-factors. For example, the matrices can be either fixed (supervised NMF, i.e. unchanged during estimation), adaptive,

or partially adaptive (just some rows or columns take part in the estimation, while others stay fixed). By setting $\mathbf{U}_j$ and $\mathbf{P}_j$ to a fixed identity matrix one obtains the original NMF model. By constraining $\mathbf{G}_j$ to only have a single nonzero element in each column one can account for GMMs and HMMs [7]. Other typical constraints are sparsity, often modeled with the L1-norm $\lambda||\cdot||_1$, which can be applied to all the sub-factors respectively, and smoothness, mostly applied to the time-varying weights $\lambda \sum_{j=1}^{J} \sum_{k=1}^{K} \sum_{n=2}^{N} (h_{j,k}[n] - h_{j,k}[n-1])^2$ to achieve temporal continuity. Constraints, like sparsity and smoothness, can simply be added to the minimization problem, where λ is a hyperparameter reflecting the strength of the constrain.

Together with the spatial covariance, the model can be represented as the set of its parameters: $\theta = \{\theta_j\}_{j=1}^{J} = \{\bar{\mathbf{R}}_j, \mathbf{E}_j^{excit}, \mathbf{U}_j^{excit}, \mathbf{P}_j^{excit}, \mathbf{G}_j^{excit}, \mathbf{E}_j^{filt}, \mathbf{U}_j^{filt}, \mathbf{P}_j^{filt}, \mathbf{G}_j^{filt}\}$. It is obvious that estimating model parameters in the maximum likelihood (ML) sense would not lead to any consistent estimation, because there are more free parameters in θ than data samples in $\bar{\mathbf{X}}$ [7]. The optimization starts by defining a likelihood criterion, which describes the probability of the observed data $\bar{\mathbf{X}} \in \mathbb{C}^{F \times N}$ given the current model parameters θ:

$$\hat{\theta} = \arg\max_{\theta} p(\bar{\mathbf{X}}|\theta) \tag{2.22}$$

In the case of prior knowledge, for example, a spatial covariance prior, an *a posteriori* criterion can be used instead, which describes the probability of the current model parameters θ given the observed data $\bar{\mathbf{X}}$.

$$\hat{\theta} = \arg\max_{\theta} p(\theta|\bar{\mathbf{X}}) = \arg\max_{\theta} p(\bar{\mathbf{X}}|\theta) \underbrace{p(\theta)}_{\text{prior}} \tag{2.23}$$

Also, Bayesian inference can be used as the criterion, whereas the former two objective functions find the estimate of θ that maximizes the likelihood function (or the posterior distribution) of θ, the goal of Bayesian inference is to infer the posterior distribution of θ, given an observation $\bar{\mathbf{X}}$ [32]. To maximize the likelihood function either a Majorize-Minimization (MM) algorithm or a Generalized Expectation-Maximization (GEM) algorithm (a specific case of MM) can be used. In the case of the Bayesian objective Variational Inference (VI) can be applied [32].

The GEM algorithm provides an iterative solutions for parameter estimation when it is difficult to optimize the ML objective in a closed-form. It consist of alternating Expectation steps (E-step) and Maximization steps (M-step). Besides the observed data $\bar{\mathbf{X}}$ and the model parameters θ, the GEM framework requires some sort of latent data Z. The choice for Z can vary. One obvious choice for Z are the source signals $\mathbf{s}_{j,f,n}$ but others can be defined too [19]. After the model and the likelihood function are defined the GEM algorithm estimates the model parameters by iterating through the following E- and M-steps with iteration index l:

1. E-step: Compute the expectation of the so-called natural (sufficient) statistics, given the observations $\bar{\mathbf{X}}$ and the current parameters θ^l.

2. M-step: Given the expectation of the natural statistics, update the parameters $\theta^{l+1} \leftarrow \theta^l$ to increase the conditional expectation of the modified log posterior.

Depending on the model and choices for the GEM algorithm the parameter update in the M-step might not be possible in a closed form and also contains an iterative optimization itself [19]. The GEM algorithm can vary in the choice of the latent data Z, the choice of the M-step updates, the initialization of parameters θ, and the stopping criterion. During the model estimation, simulated annealing can be used by leaving the last source J open to a noise term which is usually decreased in its variance during the iterations [19]. The full derivation for the GEM estimation using multi-channel NMF with the LGM assumption from equation 2.17 can be found in [7]. An overview of different variants of the GEM framework for multi-channel NMF can be found in [19]. An example for the MM and VI based estimations methods can be found in [32].

Similar to the single-channel NMF, the final sources signals are separated after the model parameters are estimated through a multi-channel Wiener filter, where $[...]^{-1}$ denotes the matrix inversion:

$$\mathbf{s}_{j,f,n} = \bar{\mathbf{R}}_{j,f,n} v_{j,f,n} \left[\sum_{i=1}^{J} \bar{\mathbf{R}}_{i,f,n} v_{i,f,n} \right]^{-1} \bar{\mathbf{x}}_{f,n} \tag{2.24}$$

The multi-channel NMF framework is not only very flexible, but it can also be seen as a *"statistical implementation of Computational Auditory Scene Analysis (CASA) principles, whereby primitive grouping cues and learned grouping cues are simultaneously used to segregate the sources, thereby avoiding error propagation due to sequential use of grouping cues. Examples of primitive grouping cues accounted by this model include harmonicity, spectral smoothness, time continuity, common onset, common amplitude modulation, spectral similarity and spatial similarity"* (in [7], page 7).

Other Methods

Besides the ICA and NMF model there exist a large variety of separation models, which are either specific implementations of the formerly mentioned models or unique approaches to the audio source separation problem. This section will present some of these approaches and their core ideas. However, it will not cover a detailed mathematical description of each method due to the large variability of models.

One of the earliest methods of music source separation uses the harmonic source-filter model, which expresses a source signal as a harmonic excitation signal filtered by a transfer function, which is considered smooth in the frequency domain. In 1973, Miller proposed to use the homomorphic vocoder to separate the excitation function and impulse response of the vocal tract [33]. This idea gave rise to many different re-synthesis models, which decomposes the sources with a set of sine waves of varying frequency and amplitude. To separate a source, one needs to estimate the fundamental frequency through some pitch detection algorithm and re-synthesis the source with an appropriate spectral envelope through a sinusoidal bank. The re-synthesis models mostly vary in their way of calculation of the fundamental frequency, clustering of the harmonic structure, estimation of the phase, and compensation for the residual non-sinusoidal noises. It suffers from a very artificial sound quality, due to the discrepancies between the estimated model parameters and the ground truth. Instead of re-synthesizing the sources one can estimate the source by filtering out all other components. In a similar fashion as the re-synthesis approach one can estimate the predominant fundamental frequencies and cluster the TF bins accordingly and then filter out the lead source (mostly the vocal) from the rest [34]. The harmonic filtering approach has a more natural sound quality, though most sound sources do not follow a strictly harmonic sinusoidal model and also contain in-harmonics and noise-like components. Also, it depends on the quality of the pitch detection method, which especially in polyphonic mixtures without a predominant source leads to wrong results. Instead of a pitch detection algorithm, prior knowledge can be used in form of a score or a MIDI file [35].

As already seen in the RPCA and NMF/NTF model, one way of separation lies in reducing the redundancy in the mixture signal by finding good low-rank approximations, which in many cases yield estimates of the source signals. Another way of utilizing this redundancy is by exploiting the repetitive nature of most popular music. Rafii et al. used the self-similarity matrix of the TF representation to identify repeating sequences of a song. By averaging over the most similar frames through a median filter they were able to filter out the non-repetitive lead instrument/vocal from the repetitive accompaniment [36]. This approach was generalized through so-called Kernel Additive Modelling (KAM), where the repeating TF bin pattern is modeled using specific kernels, which account for periodicity, self-similarity and stability over time or frequency [37].

These methods suffer from the fact that only the lead instrument or lead vocal can be separated from a repetitive background, which limits the use of these methods.

The conventional NMF model only uses spectral information but no time-depended information (or only very simple temporal continuity constraints). An extension is the incorporation of Hidden Markov Models (HMM). HMMs can model the relationship between the hidden states at different time frames, in the form of temporal dynamics [38]. Similar to the supervised NMF, the HMM structure has to be learned through isolated source signals in advance. After fitting the model to the training data, there are J models consisting of their own NMF dictionary and HMM model to represent each source [38]. This approach can also adapt to the multi-channel case through a spatial covariance structure [39].

Besides the well know Gaussian Mixture Models (GMM), which is also covered as a specific case of multi-channel NMF framework, there are other clustering algorithms used for audio source separation. An early approach by Bach et al. utilizes spectral clustering algorithms. They propose a learned similarity matrix, made out of different features inspired by Bregmans Auditory Scene Analysis, which describes the similarity of every TF bin in the mixture spectrogram. They then use the eigenstructure of a similarity matrix to partition points into disjoint clusters to achieve a binary separation [40].

As already mentioned earlier TF separation methods often arbitrarily discard phase information. As a result, the phase of each source spectrogram is unknown, and instead, the mixture phase is used, corrupting the reconstructed sources. A recently proposed separation model by Liutkus et al. utilizes Gaussian Processes (GP) to perform separation in the time domain [41]. GPs are probability distributions over functions [42]. These functions describe the source signals in the separation model and contain a mean and a covariance function, also called the kernel. Through these kernels, specific prior knowledge over the source signals can be included in the separation process. By estimating the best fitting GP in a maximum likelihood way (or an approximation of it) the source signals can be recovered. Alvarado et al. provided a way to reduce the computational complexity of the GP in [42].

2.2. Deep Learning Approach

Many model-based separation methods make assumptions about the mixing model of the sources, especially assuming a linear mixing process, and therefore do not take into account the effects of dynamic range compression and non-linear distortions. Commonly, methods yielding extremely good performance for linear generated mixtures completely break down when tested on real-world musical recordings [5]. Data-driven approaches have a clear advantage here, as they avoid making any assumptions and rather let the model be learned from a large and representative database of examples. Especially Artificial Neural Networks (ANN) have gained widespread attention over the last two decades, due to the available large computing power as well as advances in machine parallelism (e.g., cloud computing, GPUs, or TPUs). Current networks contain millions of trainable parameters, sometimes also referred to as Deep Neural Networks (DNN), and are trained to learn from a massive amount of data. For the recent break-through of DNNs, the availability of ImageNet[1], a database of 14 million (2021) labeled images used in computer vision, was a major factor. This success has also taken over other areas like signal and audio processing, often outperforming traditional methods. Also in audio source separation the classical model-based methods, like NMF models, have been outperformed by deep learning models [43].

Providing a full theoretical introduction into DNNs is out of the scope of this thesis, particularly because the strategies used to create a DNN are extremely problem/data specific, generalize just to a certain extend, and are often based on empirical studies. A short introduction is given here, for more information the MIT Press Deep Learning textbook [44] is recommended.

In general ANNs/DNNs are collections of artificial 'neurons', functions that contain trainable parameters, which are combined/cascaded together into a network structure. One of the simplest artificial 'neurons' is the perceptron, already proposed in 1958 [45]. A perceptron takes in N inputs x_n, multiplies each input with a specific weight w_n, and then sums them up, finally adding a bias b to the sum to create a scalar output. The whole process can be written in vector notation, with input vector $\mathbf{x} \in \mathbb{R}^N$, trainable weight vector $\mathbf{w} \in \mathbb{R}^N$, and trainable bias b:

$$y = \mathbf{w}^T\mathbf{x} + b \tag{2.25}$$

Next, many perceptrons are combined into a layer, where all perceptrons receive the same input but contain their own weights and biases. The whole effect of one layer on the input can be written down as a linear transformation:

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} + \mathbf{b} \tag{2.26}$$

With $\mathbf{W} \in \mathbb{R}^{N\times D}$, being the stacked weight vectors of all perceptrons in that layer, D

[1] `http://image-net.org/about-overview`

being the number of perceptrons, and $\mathbf{y} \in \mathbb{R}^D$ being the stacked output of all perceptrons in that layer. These layers are then cascaded into a so-called Multi Layer Perceptrons (MLP). Since the combination of several linear transformations can be described by a single linear transformation (i.e. making the combination of several layers equal to a single layer), the output of each layer is processed through a non-linear function, also called the activation function, before it is further processed by the next layer. There are lots of different non-linear functions used in deep learning, each of them containing different properties. Most well know are the Rectified Linear Unit (ReLU), the Sigmoid function, and the TanH function. MLPs are also called Feed-forward Neural Networks (FNN), as the information moves only forward, from the input layer to the output layer. The number of layers L determines the depth of the network. Layers apart from the input and output of the network are called hidden layers. Most DNN architectures are designed according to the information bottleneck principle [46], reducing the data dimensionality on every layer, forcing the network to only retain the most relevant information. In general deeper networks are used to solve more complex tasks, as they can re-process and re-structure the input data and find latent structures inside of it. A very common problem with too large networks (or too small datasets), is overfitting. Here, the network has too many free parameters and rather memorizes the whole training set instead of finding abstract latent structures in the data. This leads to a bad generalization (i.e. bad performance on unseen data).

DNNs are optimized using an objective function $\mathbb{L}(\mathbf{X}, \mathbf{Y}, \theta)$, which takes in the input data $\mathbf{X}$, the target labels $\mathbf{Y}$, and the trainable network weights θ. In most cases, the objective function describes some kind of distance between the network output and the desired output, so the goal is to minimize the objective function:

$$\hat{\theta} = \arg\min_{\theta} \mathbb{L}(\mathbf{X}, \mathbf{Y}, \theta) \tag{2.27}$$

Due to the non-linearities applied after each layer, DNNs cannot be optimized in a closed-form. Instead, training is achieved by stochastic gradient descent [47]. Here a randomly chosen small set of data-points called a mini-batch is used to estimate the gradient and optimize the network parameters θ in an iterative way:

$$\theta^{\text{new}} = \theta^{\text{old}} - \mu \nabla_{\theta} \sum_{i=1}^{B} \mathbb{L}(x^i, y^i, \theta^{\text{old}}) \tag{2.28}$$

Where B is the mini-batch size and $\mu \in \mathbb{R}_+$ is the learning rate, a small scalar, which determines the size of the gradient step. SGD is much more computationally efficient than normal gradient descent, as the gradient only needs to be calculated over a small set of data points, instead of the whole training data. For the calculation of the gradient, the back-propagation algorithm is used [48], which is an efficient recursive implementation of the chain rule. There are a variety of more sophisticated algorithms to apply the gradient than SGD like Adam which uses adaptive learning rates for each parameter in the network [49].

Besides the simple feed-forward structure of MLPs, many different neural network structures and layer types exist. The two most important are the Convolutional Neural Networks (CNN) and the Recurrent Neural Network (RNN). CNNs are based on convolving[1] their input with many small learnable kernels, also referred to as filters, in place of matrix multiplication. These kernels are much more parameter efficient, as they only consider the local structure of the data and share the kernels across the whole data. A very common example of such a learned kernel is an edge detector. The filtered output of a convolutional kernel is called a feature map. CNNs require the input data to have a grid-like topology, where a data point inside the grid shows some correlations to the points surrounding it. Examples are time-series data, which can be thought of as a 1D grid taking samples at a regular sampling-rate, and images, which have a spatial structure as a 2D grid of pixels [50]. Therefore both one- and two-dimensional convolutions are mainly used in CNNs depending on the input data structure. Regardless of the dimensionality of the data, a convolutional layer always contains an additional feature dimension, which size is determined by the number of kernels used inside that layer. Similar to FNNs, CNNs are arranged by cascading layers together, each one followed by a non-linearity. After the non-linear function, a pooling function is often used to down-sample the learned feature maps, reducing the size of the data and compressing its relevant information. Instead of a pooling function, the size can also be directly reduced by using strided convolutions, which simply skips every n-th value. Due to their local operations and parameter efficiency CNNs are less invariant to scaling and translations, as well as less prone to overfit [50]. They also make use of parallelization very efficiently. RNNs, on the other hand, use internal feedback connections to process sequential data. In contrast to FNNs and CNNs, they contain cycles in their graph and compute the output for a sequence step from both the input at that step and their hidden state from the previous step [51]. For offline applications, bidirectional RNNs can use a second recurrence in reversed order, increasing the receptive field also to future steps of the sequence. RNNs are the classical way of processing data with a temporal structure, as they can model the dynamics of such data. Since they process each step of the sequence one after another they cannot be parallelized efficiently and as a consequence take longer to train. Another issue with the recurrent feedback connections inside RNNs is the exponential growth/decay of their hidden states and their respective gradients. This is called the vanishing and exploding gradient problem and makes it difficult to train standard RNNs. Modern RNNs, therefore, use more complex 'neurons' such as the Long-Short-Term Memory unit (LSTM) [52] or the Gated Recurrent Unit (GRU) [53], which encapsulate their recurrent hidden states through gating mechanisms to alleviate gradient problems. Recent trends show that CNNs are as efficient as RNNs in processing time-series data and become more and more popular, especially in audio processing, due to the large time series data involved. Also, Convolutional Recurrent Neural Networks (CRNN) are used more and more. Here, convolutional layers extract local information, reducing the high dimensional data to a compact set of features and recurrent layers then model the dynamics of these features [54].

[1] Most implementations are actually based on cross-correlation, instead of real convolutions.

Another alternative to RNNs becoming increasingly popular are Attention-based models which learn alignments through weighted similarities between the input and output sequences or within the input sequences jointly with the target optimization [55].

Audio Source Separation can be seen as a sequence to sequence regression problem, where both input and output are values in a continuous range [43]. Many architectures were proposed in the past, such as FNNs, CNNs, and RNNs with both LSTMs and GRUs as well as combined architectures. Most deep learning source separation methods typically estimate masking operations in the TF domain as they were originally adapted from the NMF framework [56]. But also waveform-domain methods have become more and more popular recently [57]. The next sections will provide an overview of the relevant network architectures, training methods, and available datasets and finally point out problems of current deep learning methods, which will be approached in chapter 3. For a more general overview of deep learning methods used in audio processing, the summery of Purwins et al. is recommended [43].

Datasets

One downside of DNN based source separation is the need for a large amount of training data. Most audio-related deep learning tasks, except speech recognition, face relatively small datasets, already limiting the complexity of these deep learning models [43]. This is also a big problem in music source separation because data acquisition is a difficult task. A data point has to contain the final mix down as well as all isolated source tracks. In speech source separation this task is quite easy to solve, as in general one can take any isolated speech recording and mix them together to produce a new data point. There are large corpora of speech recordings already available, for example, the Wall Street Journal (WSJ1) data set, containing over 78,000 utterances[1]. Besides that, also the generation of new data sets is quite straightforward. In the case of music, this generic way of constructing new data is not possible since different sources in a song are highly correlated and cannot be mixed together in randomly. They are subject to complex harmonic as well as long- and short-term temporal relationships and span a very small and specific subspace in the ways they're mixed together. By mixing random musical phrases of different pitches and tempos together, one would certainly never obtain a common popular music piece. Therefore, professionally composed, recorded, and mixed songs have to be used for training [5]. Even though there is an endless amount of these mixdowns (e.g. the Spotify database containing over 60 million songs[2]), it is almost impossible to get access to the underlying isolated source tracks, as they are considered amongst the most precious assets of their right holders. Because of these copyright issues, most datasets are either private or comparably small and specific [43].

[1] `https://catalog.ldc.upenn.edu/LDC94S13A`
[2] `https://newsroom.spotify.com/company-info/`

Common public datasets used for music source separation are:

1. IKala and MIR-1k datasets - both used for vocal/accompaniment separation and containing around 2 hours of music separated into two source tracks [58][59].
2. Medley-DB 2.0 dataset - created for a variety of MIR tasks. It contains by now around 200 tracks of music, most of them including the isolated source tracks, split into 54 categories [60].
3. DSD-100 and the MUSDB18 - both specifically created for multi-instrument music source separation, within the SigSep Evaluation campaign. The MUSDB18 is an extension of the DSD-100 and contains 150 tracks, separated into 4 isolated tracks: vocals, bass, drums, and other. It was created partly with the Medley-DB dataset [61] [62].
4. Slakh2100 - created to account for the relatively small sizes of the other datasets. It contains 2100 tracks yielding a total amount of 145 hours of music, categorized in 34 different instruments. The dataset was generated with public available Midi scores played by a sampler. It does not contain any vocals and due to the generative midi procedure only partly resembles a real song [63].

Since there is only limited data available, a common method during the training of a DNN is data augmentation. For music source separation there are no standardized augmentation procedures and they vary from one method to another. The most common data augmentation techniques for music source separation were proposed by Uhlich et al. in [64] and again evaluated Pretet et al. in [65]. They suggest to modifying the audio data by:

1. random swapping of left/right channel for each source
2. random loudness re-scaling of the source tracks
3. random time-stretching
4. random pitch-shifting
5. random filtering
6. random mixing of source tracks from different songs

The implementations and results of these data augmentation techniques vary. Peréte et al. evaluated each method separately and discovered that only channel swapping, pitch-shifting and time-stretching contribute to the performance and noted a overall maximum performance gain of $+0.2\,\mathrm{dB}$ SDR, the other methods did not contribute significantly [65]. Uhlich et al. results are very similar to this[64]. Both data augmentation techniques were only evaluated under the vocal/accompaniment separation setting and not in a multi-instrument separation setting. A very unique augmentation method was proposed by Stoller et al. in [66], where they use another adversarial discriminator networks that is trained to distinguish real recordings from separator samples, making it possible to implicitly train the separator network on unlabeled data.

Time-Frequency Masking Separation

Since NMF-models had shown that source separation is particularly efficient in the TF-domain, using primarily the spectral information for separation, most deep learning approaches also operate in the TF-domain.

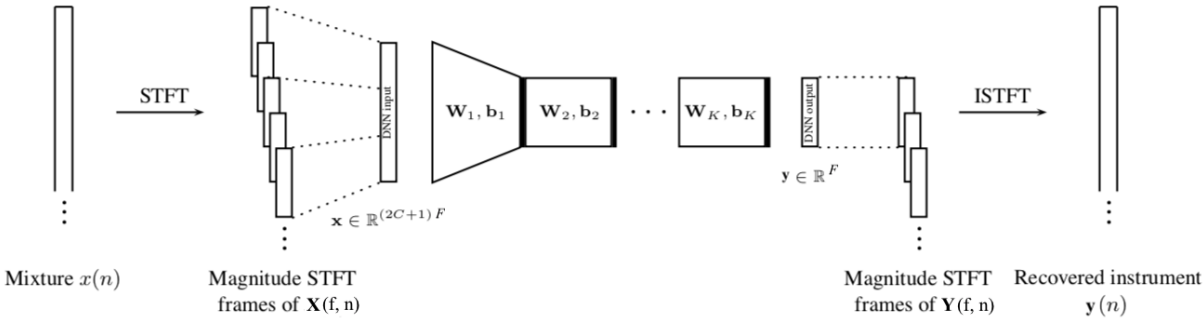

Figure 2.2.: FNN architecture for music source separation (Adapted from [67]).

One of the first to use a DNN for music source separation was Huan et al. [68]. They proposed a classical RNN structure with 3 hidden layers of 1000 hidden units each, processing three concatenated STFT frames at a time to predict one separated STFT frame and showed that this recurrent architecture outperforms the simple feed-forward architecture. At the same time, Uhlich et al. developed a similar approach using a FNN whose input consists of five concatenated STFT frames, trained in a consecutively way, adding one layer at a time [67], up to five layers in total. Figure 2.2 shows the proposed architecture. Later Uhlich et al. also updated their architectures with bidirectional recurrent LSTMs, improving their results over their previous feed-forward architecture [64].

Both these approaches have a relatively small temporal input context of only a few consecutive STFT frames and instead use the recurrent structure to model longer temporal structures. Simpson et al. proposed a different approach, using a much larger FNN with an input context of 20 consecutive STFT frames concatenated to a super-vector. The network had three layers and an output of 20 consecutive STFT frames. They used a sliding window approach with a hop-size of one frame to get 20 estimates for every STFT frame and then took the average over all estimates to get the final source spectrogram. Since the FNN processed the mixture spectrogram with a sliding window, they called their model a convolutional DNN, as it could be also seen as a CNN with a kernel size of 20×1025[1]. However, their model does not utilize the parameter efficient structure, which 'modern' CNNs provide, and it contains over one billion parameters (!) [69].

[1] 1025 being the number of frequency bins in the STFT.

Chandna et al. pointed out that these DNN architectures are not completely exploiting the local TF structure and instead rely on global features across the entire frequency spectrum [70]. They were the first to adapt a 'modern' CNN design and an informational bottleneck structure to the music source separation problem. They used two convolutional layers to encode 25 frames of the mixture spectrogram, two feed-forward layers as the bottleneck, and another two transposed convolutional layers as the decoder and showed that this parameter efficient design performed as well as previously proposed MLP structures while containing fewer parameters [70].

Most researchers use a relatively simple objective function to train their models, like the L1/L2-norm between the estimated spectrogram and the target spectrogram, similar to the ones used for NMF (see eq.: 2.11-2.13). These objective functions are in many cases 'good enough', an example for a more complex objective function is given in [71].

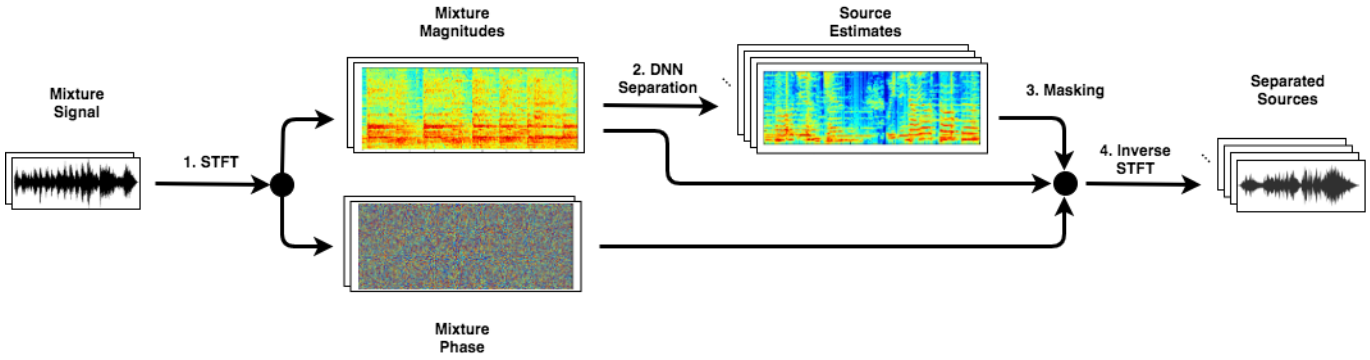

Figure 2.3.: Sketch of the TF masking separation process.

All DNNs operating in the TF-domain make use of a masking function for the final separation process, which was adapted from previous NMF-based separation approaches. The TF mask enforces the constraint that the sum of the estimated results is equal to the original mixture [68]. Figure 2.3 visualizes the spectral masking separation process. The STFT of the mixture waveform is calculated and divided into phase and magnitude information. The magnitude information is passed either as a concatenated super-vector $\mathbf{x} \in \mathbb{R}^{NFI}$ to a FNN/RNN or as a tensor $\mathbf{X} \in \mathbb{R}^{N \times F \times I}$ to a CNN, where N is the number of included STFT frames, F is the number of frequency bins in the STFT and I is the number of channels (1 = Mono, 2 = Stereo). In general, N has to be chosen carefully to provide a large enough temporal context but prevent too large amounts of parameters as well as memory overflow. The DNN then yields a source estimate $\mathbf{Y}_{j,k} \in \mathbb{R}^{N \times F \times I}$ for each frame-block of length N in a song, which is either simply concatenated or concatenated by an overlap-and-add procedure. The concatenated source estimates are used to create a final TF mask for each source track, which filters the complex-valued original mixture STFT. The obtained filtered spectrograms are then used to reconstruct the time domain signals through an ISTFT.

There are several masking functions, which can be used to filter the mixture STFT. The most rudimentary approach is the binary mask, which simply assigns the TF-bin to the source estimate with the highest energy:

$$\hat{\mathbf{S}}_j = \mathbf{M}_j \odot \bar{\mathbf{X}} \qquad \text{with} \qquad \mathbf{M}_j = \begin{cases} 1 & \text{if } \frac{\mathbf{Y}_j}{\sum_{i=1}^{J} \mathbf{Y}_i} > \frac{1}{J} \\ 0 & \text{else} \end{cases} \tag{2.29}$$

The binary mask comes with the drawback of a characteristic 'musical' noise, due to the abrupt phase and amplitude changes in the spectrogram and is mostly outdated [72]. Except Simpson et al. [69], most researchers have focused on a soft masking strategy, where the TF mask lies in the continuous interval from 0-1, which has the advantage of strongly reducing musical noise:

$$\hat{\mathbf{S}}_j = \sum_{j=1}^{J} \frac{(\mathbf{Y}_j)^{\alpha}}{\sum_{i=1}^{J} (\mathbf{Y}_i)^{\alpha}} \odot \bar{\mathbf{X}} \tag{2.30}$$

The hyperparameter α can either be set to 1 for a normal ratio mask or to 2 for an optimal single channel Wiener Filter [72]. Both the binary mask and the soft mask ensures that the original mixture is obtained when all separated sources are summed up again: $\sum_{j=1}^{J} \hat{\mathbf{S}}_j = \bar{\mathbf{X}}$. Since they both only contain real-valued numbers between 0 and 1 they automatically assign the mixture phase to every separated source, which especially affects transient and sounds of low frequency negatively.
Most of the approaches mentioned above consider the single-channel source separation. Nugraha et al. adapted the GEM algorithm, used for the multi-channel NMF/NTF (see section 2.1), to DNN based separation models, by replacing the NMF/NTF-model with several DNNs. The DNNs are used to estimate $y_{j,f,n}$, the spectral energy of source j at TF-bin f, n, while the GEM model predicts the frequency-dependent spatial covariance matrix $\bar{\mathbf{R}}_{j,f} \in \mathbb{C}^{I \times I}$ in an iterative way. Their DNNs architectures are very similar to the one from Uhlich et al. shown in figure 2.2. Though, instead of using one DNN for the whole estimation, they use several DNNs. Each DNN is used at a different iteration stage of the GEM algorithm to re-estimate the source energies based on the previously estimated energies and spatial covariances [56]. The final MWF mask therefore also includes complex-valued spatial information of the multi-channel mixture $\bar{\mathbf{x}}_{f,n} \in \mathbb{C}^{I}$:

$$\hat{\mathbf{s}}_{\mathbf{j,f,n}} = y_{j,f,n} \bar{\mathbf{R}}_{j,f,n} \, [\sum_{i=1}^{J} y_{j,f,n} \bar{\mathbf{R}}_{i,f,n}]^{-1} \; \cdot \; \bar{\mathbf{x}}_{f,n} \tag{2.31}$$

The adaptation of MWF into DNN based source separation was shown to be beneficial for the separation quality [56]. The iterative GEM algorithm to derive the optimal MWF coefficients was implemented in a python toolbox[1] in the course of the Signal Separation Campaign, to provide an easily accessible MWF masking function for any DNN architecture.

[1] `https://github.com/sigsep/norbert`

All previous DNN separation approaches directly estimate either the source signals or a source mask, using either one network per source class or a single network with multiple dedicated source outputs. This makes it impossible to separate signals in a class-independent way, where the included source classes are not known in advance. Hershey et al. proposed a different method where they trained a deep network to produce a spectrogram embedding that is discriminative for grouping labels calculated from the training data. The source separation is therefore implicitly encoded in the learned embeddings and can be "decoded" with a clustering algorithm. The approach is inspired by previous clustering-based separation models and is closely related to the perceptual grouping theory of computational auditory scene analysis [73]. They used a bidirectional LSTM network to model the embedding process. Instead of using an objective function, which minimizes the distance between the networks estimate and the original source spectrogram, they proposed a novel class-independent objective function. The network takes in the mixture spectrogram $\mathbf{X}$ and outputs a D dimensional embedding vector $\mathbf{v}_{f,n} \in \mathbb{R}^D$ for every TF bin in $\mathbf{X}$, so that the embedding features of the TF regions dominated by the same source are forced to get close to each other and those dominated by different sources are forced to get separated from each other [74]. The embedding vectors are stacked into a matrix $\mathbf{V} \in \mathbb{R}^{FN \times D}$ and the network is trained to minimize the following objective function:

$$\begin{aligned} \mathbb{L}(\mathbf{V}, \mathbf{Y}, \theta) &= ||\mathbf{V}\mathbf{V}^T - \mathbf{Y}\mathbf{Y}^T||_F^2 \\ &= ||\mathbf{V}^T\mathbf{V}||_F^2 - 2||\mathbf{V}^T\mathbf{Y}||_F^2 + ||\mathbf{Y}^T\mathbf{Y}||_F^2 \end{aligned} \tag{2.32}$$

In this case, $\mathbf{Y}\mathbf{Y}^T$ is an $FN \times FN$ binary affinity matrix, where the element is given by $\mathbf{Y}\mathbf{Y}^T[n, \hat{n}] = 1$ if TF bins n and $\hat{n}$ are dominated by the same source, otherwise the element is set to 0. The objective function encourages the mapped embedding vectors to become parallel if they are dominated by the same source and orthogonal otherwise [73]. The second line of equation 2.32 prevents the objective function to calculate the $FN \times FN$ affinity matrices and makes it possible to compute the gradients of the network parameters θ within a reasonable amount of computational effort. After training the network, the embedding vectors are used with computationally efficient clustering algorithms such as k-means, to assign each TF-bin of the mixture spectrogram a binary mask for separation. Since the number of clusters in the embedding space can be varied, the network can separate mixtures with arbitrary amounts of source signals involved. Hershey et al. proved this by separating three speaker mixtures with a network only trained with two speaker mixtures [73]. Li et al. improved the results achieved by Hershey et al. using a CNN instead of the proposed RNN [74] and Lou et al. adapted the deep clustering method to music source separation [75].

Although Deep Clustering is able to separate arbitrary class-independent mixtures, it can only separate through a binary TF mask, which has been shown to be less effective than other masking types [72].

Modern deep learning frameworks, like Tensorflow and Pytorch, made the development of DNNs for source separation easier and led to a quite large variety of DNN architectures. Jansson et al. adapted the fully convolutional U-Net architecture, initially developed for medical imaging [76], for the task of source separation, because of its proven capacity of recreating the fine low-level detail, required for high-quality audio reproduction [77]. The U-Net architecture is formed by an auto-encoder architecture with additional skip-connections between layers at the same hierarchical level in the encoder and decoder stage. While normal "hourglass" auto-encoder architectures are successful in discovering global hidden structures in the data, a lot of local details are lost in the bottleneck compression. The skip-connections alleviate this problem by allowing high-resolution details to flow directly beside the deep encoded information inside the bottleneck.

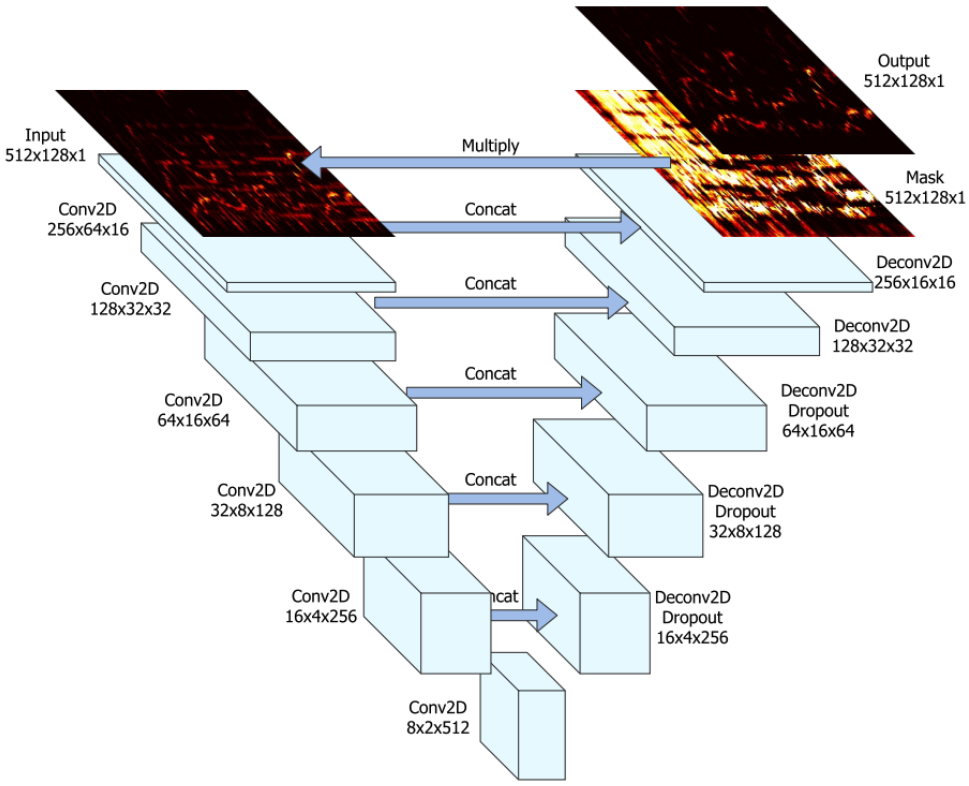

Figure 2.4.: Fully convolutional U-Net architecture for music source separation (Taken from [77]).

Figure 2.4 shows the architecture proposed by Jansson et al. in [77]. The U-Net contains six encoder and six decoder stages and a final internal masking stage. They used two-dimensional convolutions with a stride of 2×2 in the down-sampling path and transposed convolutions in the up-sampling path, both with a kernel size of 5×5. Each convolutional layer is followed by a batch normalization layer, which has been shown to improve convergence during training [78], and a final leaky ReLU activation function.

They implemented three drop-out regularization layers in the deeper part of the network to prevent over-fitting [79]. The final layer is used with a sigmoid activation function to create an internal mask, which is then multiplied to the input spectrogram. The internal masking procedure was also later proven to be beneficial for separation performance in [80]. The model was trained using the Adam optimizer [49] on a large private database on the vocal/accompaniment separation task and showed excellent performance [77]. It was also adapted for multi-instrument separation tasks by a researcher team of Deezer under the name Spleeter [81]. The auto-encoder structure with additional skip-connection was reused by a lot of researchers in music source separation, like [82], [65], [80], [54], [83] and builds the current foundation of modern source separation networks.

Stoter, Uhlich et al. updated their previously proposed recurrent LSTM model with 'modern' deep learning techniques like batch-normalization [78], dropout regularization [79], bottleneck design [46], skip-connections, and an internal masking process [80] and created the open-source separation model OpenUnmix [84].

Takahashi et al. adapted the densely-connected convolutional structure from DenseNet [85] to the source separation problem and created their MMDenseLSTM architecture. The idea of DenseNet is to use concatenated output feature maps of all preceding layers as the input to all succeeding layers, creating a connection between every layer of the network. These connections effectively support the gradient flow inside the network, making it possible to train much deeper networks while also requiring significantly fewer parameters (as features maps can be reused in every layer). Since DenseNet does not allow any down-sampling paths, because of its inter-connectivity, Takahashi et al. created densely-connected blocks, which are followed by a down-sampling layer and arranged them in an encoder-decoder structure with typical U-Net skip-connections between them. They furthermore introduced a multi-band structure with dedicated dense blocks to particular frequency bands, as otherwise, most kernels in a convolution layer focus only on the higher energy band and neglect frequency regions with lower energy. By limiting the bands that share the same kernels the network can efficiently capture local patterns [86]. Although they process the whole spectrogram in multiple bands, their model only contained 0.3 million parameters, due to the densely-connected architecture. They later on updated their dense block architecture with another LSTM block in a unified architecture together with a final MWF, which achieves the current State-of-the-Art results and outperforms the Ideal Binary Mask[1] [54].

[1]The binary mask created with the original source tracks, see [14] for more information.

Time Domain Separation

Recently, the first end-to-end DNN models have been developed which directly perform source separation in the raw waveform-domain. Since the TF mask relies only on the magnitude information and discards the phase, it has a natural upper bound on the performance of all methods relying on this strategy, for instance, the Ideal Ratio Mask (IRM) or the Ideal Multi-channel Wiener Filter (IMWF) [57]. Time-domain models are not limited by this upper bound and as TF masking models start to approach the oracle performance, it is a logical step to move on to a waveform-to-waveform separation approach.

One of the first approaches to waveform domain separation was done by Stoller et al. by adopting the U-Net form Jansson et al. [77] to the waveform-domain [87]. They replaced the 2D convolutions with longer 1D time-domain convolutions and used a learned up-sampling layer instead of transposed convolutions in the decoder, to prevent aliasing. They trained a single model with multiple outputs for the multi-instrument separation task. The model was submitted to the 2018 SiSec evaluation campaign but was not able to perform as well as models using a TF masking approach.

For speech source separation, Luo and Mesgarani proposed Conv-Tasnet, in which they reuse the TF masking approach but instead of using an STFT they propose to use a learned transformation. The model operates directly in the waveform domain for both the inputs and outputs and uses a learned front-end that transforms back and forth between the input mixture waveform and an over-complete basis representation. The transformed representation is then masked by an internal separation network. They first proposed a recurrent LSTM network to estimate the mask [88] and later updated their model with stacked convolutional residual blocks [89]. The output of the encoder is multiplied by the mask obtained from the internal network before going through the decoder back to the waveform-domain. Conv-Tasnet surpasses the IRM oracle performance in speech separation. Defossez et al. adapted Conv-Tasnet to the music separation task. Although it did not outperform the IRM on this task, it outperformed all current TF masking models [57].
Furhtermore, Defossez et al. proposed Demucs, their own end-to-end source separation network architecture, which is inspired by models used for music synthesis rather than a masking approach. Their model consists of a convolutional encoder and a decoder based on wide transposed convolutions with large strides and typical U-Net skip-connections between each encoder/decoder as well as a bidirectional LSTM in its deepest stage [57]. Demucs predicts all source instruments through a single network with multiple outputs and achieves very similar results to Conv-Tasnet. They also did a perceptual evaluation study, showing that the synthesis separation approach of Demucs yields slightly higher sound quality and fewer artifacts, while the masking approach of Conv-Tasnet yields slightly less contamination by other sources but an inferior sound quality [57].

Multi-Class Separation

In general, there are three main design concepts for the separation process of numerous source classes [83]:

1. Multiple independent DNNs, one trained for each source class.
2. A single DNN with multiple outputs, using a dedicated output for each source class.
3. A single conditioned DNN, which outputs a specific source class depending on the given condition.

Multiple DNNs for multi-class separation is the most common approach used by many researchers, as it is straightforward to implement and yields good results. However, in the case of high numbers of source classes, this method scales badly in terms of computational overhead, training/separation time, as well as the number of contained model parameters. With currently available datasets, like MusDB18, which only contain a few different source classes, these issues are not so serious. Still, it is an important aspect for future research, where more complex and detailed datasets might be available. Also for applications, where the computational overhead is crucial, such as real-time and mobile applications, a single DNN architecture might be beneficial. In general, using one DNN with multiple outputs solves the problem of computational overhead quite well. It is also simple to implement, as just the last layer of the network needs to be changed. Kadandale et al. proposed the M-U-Net, which adds multiple output layers to a U-Net and showed an equal performance to dedicated U-Nets while containing fewer parameters and lower training times (by a factor of $\frac{1}{J}$) [83]. Training a single DNN to separate multiple sources might result in a bias of the model towards sources with higher energy, as their gradients can have higher magnitudes [90]. To prevent this, different weighting strategies have been proposed, which weight each source class according to their overall energy in the whole dataset (see [83] and [90]). Conditioning a DNN to extract different source classes depending on a given condition combines the negative effects of both previously mentioned strategies. The computational effort scales exactly the same as for dedicated DNNs. This is so because each data point needs to be passed through conditioned DNN multiple times in every training epoch, one time for each source class [83]. A good overview of different conditioning strategies for music source separation can be found in [91]. They will be not further explained here.

Especially for models that operate in the waveform-domain, the single DNN approach is advantageous, as they need bigger convolutional kernels and have to process much longer sequences to achieve the same receptive field. Therefore the training time can be quite large and training multiple networks might not be feasible in a reasonable amount of time. All previously mentioned waveform-domain networks use a multi-output approach. Another example is Meta-Tasnet, which is a conditioned Conv-Tasnet used for music source separation [92].

Current Problems

Although first waveform-domain models have been successfully developed, most architectures still rely on a TF mask approach. 2D convolutional U-Net architectures converge in a relatively low training time, provide a highly detailed output, are less prone to over-fitting, and are able to capture a quite large receptive field without running into memory issues. Hence, they're still the most widespread network architectures in music source separation. One major problem with the 2D convolutional processing, originally developed for image processing, when being used on a spectrogram, is often ignored - the exponential scaling of the frequency axis. While the general benefit of CNN's lies in their ability to only operate on local regions and generalize the obtained local features over the whole input data while minimizing the number of required parameters, this attribute relies on equally and linear scaled input dimensions. This applies to the time dimension of the spectrogram, as time does not change its 'speed' depending on the spectrogram region. The distance between two successive spectrogram frames always remains the same. However, the frequency dimension does not fulfill this requirement. This can be easily shown with an example: Relationships between frequencies, especially in music are based on the harmonic series, which consists of multiple frequencies of a fundamental frequency. A musical tone at a fundamental frequency of 100 Hz will have the first harmonic at 200 Hz, the second at 300 Hz, and so on. On the other hand, a tone with a fundamental frequency of 200 Hz will have the first harmonic at 400 Hz and the second at 600 Hz, and so on. Hence, the distance between a fundamental and its harmonics changes according to the frequency region. This exponential scale in the frequency axis makes it very difficult for CNN's to learn harmonic structures inside of its linear scaled kernels. A very common approach to tackle this issue is to use a transformation that results in a logarithmic or near logarithmic scaled frequency axis, like the constant-Q transformation or a Mel-scale transformation. These transformations work very well for classification tasks, where the CNN yields an abstract low dimensional output. In audio source separation, however, the output of the network is again a fully detailed spectrogram, for which an inverse transformation into the time-domain is required. Log-scale-like transformation lose a lot of detail, especially in higher frequency regions. Also, their inverse does not exist and they rather rely on approximations. Therefore, they are an unsuitable representation for audio source separation. The multi-band structure of MMDenseLSTM, which splits up the spectrum into three bands and processes them with separate band-specific CNN kernels solves this problem partly and results in State-of-the-Art separation performance, showing the benefit of tackling this issue [54].

An even more severe issue with the TF masking approach is the lack of phase information. This degrades the separation process in two ways. Firstly, the network cannot utilize the phase information to separate sources, this especially affects the performance in a multi-channel situation. Secondly, the separated sources are created with the mixture phase, which is a crude approximation of the real source signal. While the MWF is at least able to use the phase differences caused by the spatial location of the source to improve the separation, it cannot separate mixed phase information in TF-bins, which

are occupied by different sources. This 'noisy' phase affects sources differently. Vocals, for example, are relatively spares in the TF-domain and mostly occupy higher frequency regions with an already relatively dense amount of TF-bins. Therefore, the effect of the 'noisy' phase is not very audible in separated vocals, as in general, they have a high chance to occupy their own TF-bins. The effect on other instruments is much higher. The bass mostly occupies regions of low frequency. Here the TF-bin density is low, due to the nature of the STFT, and many frequencies share the same TF-bin. Therefore, also the phase information is much more likely to be corrupted by different sources. This is can be perceived when listened to the bass sources separated with a TF masking approach. The sound is very diffuse and blurred. Interestingly, this can also be seen in the evaluation results when comparing TF domain models to waveform domain models. The evaluation results of the bass are usually much worse than the results of the vocals with a TF masking model, while they're almost identical with a waveform-domain model (see table 1 in [57] as an example). Also, the drum separation is affected by the 'noisy' phase. Here, this is not so visible in the evaluation results, as drums are separated quite well with the TF masking approach. However, drums contain a lot of transient sounds with impulse-like attacks. These impulses require the phases of the whole spectrum to align specifically. This does not happen with the 'noisy' mixture phase and the separated drum signal contains very hollow transients, which can be heard quite well and even seen (shown in figure 4.4).

In the next chapter these problems are approached by modifying existing DNN architectures and expanding their contextual perceptive field to music specific properties. A way to provide the CNN kernels access to the harmonic series is developed and evaluated. To align the 'noisy' mixture phase, another post-processing network, operating in the time-domain, is created, which takes advantage of the repetitive nature of music. Furthermore, different data augmentation, as well as data generation strategies are developed and evaluated to counteract the general data scarcity.

3. Method

The practical part of this thesis deals with existing CNN based TF mask separation methods, due to previously mentioned advantages, especially their relatively low convergence time. Furthermore, it is a well-researched topic with lots of information about network architectures and hyper-parameters available. The goal is to evaluate current methods and find extensions, which solve the problems pointed out in section 2.2. The implementation of all experiments is done with Python and Tensorflow.

3.1. Dataset

Throughout this thesis, the MusDB18 dataset will be used, as it is already structured for the source separation problem and also serves as the standard evaluation baseline for most music source separation papers. Therefore, the target will be to extract the four source tracks including vocals, bass, drums, and others (i.e. residual instruments like guitar, piano, ..., mixed into one track) from a professional stereo mixdown. To speed up the training procedure all tracks are re-sampled from their original sample rate of 44.1kHz to 16kHz. This is a common practice [84]. The provided split of 86 songs for training, 14 songs for validation, and 50 songs for evaluation is used. The STFT is performed with a window size of 2048 samples and a hop size of 512, these parameters showed the best results in my initial experiments. The songs are sliced into segments of $\hat{N} = 512$ STFT frames, resulting in a length of 16.5 seconds per data point. 32 slices are sampled randomly from each song, which prevents a bias towards longer songs. This sampling technique showed better results than slicing up all songs throughout. From the STFT slices, the highest frequency bin is dropped to reduce the frequency bins to 1024, which has a much better down-sampling property for CNNs (i.e. power of 2). After the separation, the dropped frequency bin is replaced by the mean value of all other frequency bins in the respective frame. One data point consists of the magnitudes of the mixture slice and the corresponding magnitudes of the four source slices. The phase information is discarded during the training process. The magnitude values are turned into a log scale, clipped between 0 and -85 dB, and compressed to a single byte per value. This reduces the amount of required disk space and was almost not audible when being decompressed. The compressed data points are saved in the Tensorflow internal '.tfrecord' format to obtain a fast data input pipeline. This process avoids long audio file loading times as well as expensive STFT calculations during training, effectively reducing the training time. The final training set contains 2752 data points and the validation set contains 448. During training, the magnitude values are converted back to linear scaled 32bit float values.

3.2. Data Augmentation

The used data augmentations are inspired by the ones used in [65] (see list 2.2). Pretet et al. performed their augmentations on the STFT magnitudes frames[1]. I assume that this results in a non-optimal augmentation, as these are rough approximations compared to their time-domain implementation, especially for the time-stretching and pitch-shifting augmentation techniques.

Therefore a completely time-domain-based augmentation setup is used in this thesis. Channel swapping per source is performed with a chance of 50%. The loudness of each source track is re-scaled with a uniformly distributed random factor between 0.5 and 1.25. A novel filtering augmentation is introduced, where every instrument is randomly filtered with a chance of 50%. The filtering is achieved through a convolution of the original source track with a randomly created normalized Impulse Response (IR) with a randomly selected length between 2 and 32 samples. The IR has a high chance to have a high value in its first sample. This will create a new frequency response, which still preserves some of the frequency relationships of the original source. Instead of using dedicated pitch-shifting and time-stretching algorithms, which in general degrade the audio quality, the original audio is simply re-sampled to a lower or higher sample rate and then played back in the original sample rate. This simultaneously results in a pitch shift and time stretch without introducing artifacts into the audio. The re-sampling factor is calculated by $2^{(u/12)}$, with u being drawn from a uniform random distribution between -6.0 and 6.0, spanning one octave. The re-sampling is applied to all source tracks with the same factor to preserve rhythmic and harmonic relationships. The modified source tracks are then mixed together to create a new mixture. To simulate the non-linear mixing process done by a professional audio engineer during 'mastering', a soft non-linearity is applied to the new augmented mixtures, with: $\hat{\mathbf{x}} = tanh(\lambda \mathbf{x})$. Here, λ is a scaling factor, which ensures that the non-linear distortion is just barely audible. The augmented dataset contains three times as many data points as the original dataset and also saved in a '.tfrecord' dataset, using the same procedure as described before.

Although always taken as a fact in most literature, to my knowledge no one ever evaluated the assumptions that the high harmonic and temporal correlation between instruments in a music mixture is necessary to the training success of a DNN for music source separation. Therefore, a second dataset is created in which the mixtures are created by randomly mixing four different source tracks from any of the available songs in the MusDB18 training split. It contains three times the amount of data points as the original dataset to take advantage of the combinatorial possibilities of this data generation technique. Both datasets will be used for training to evaluate if musical correlations between different source tracks inside the training data are really necessary for the separation success of a DNN. Besides this evaluation, the randomly generated dataset is also used as another data augmentation source.

[1] This can be seen in their online published models at `https://github.com/deezer/spleeter`

3.3. Training

If not mentioned otherwise all experiments utilize one specific DNN per source, yielding four separate training's. One training consists of 60 epochs per DNN, where one epoch equals an iteration over 3200 data points. When using data augmentation, the augmented data points are randomly interleaved between the original data points, and the iterations per epoch are kept fixed at 3200 for a fair comparison. Before every epoch the complete dataset is shuffled once. After 60 epochs the model at the epoch with the best validation loss is evaluated. The L1 distance between the estimated source $\mathbf{Y}$ and the original source $\mathbf{S}$ is used as the loss function during training:

$$L(\hat{\mathbf{Y}}, \mathbf{S}) = \frac{1}{FNC} \sum_{f,n,i}^{F,N,I} ||\mathbf{Y}[f,n,i] - \mathbf{S}[f,n,i]||_1 \tag{3.1}$$

The L1 loss has been used in many source separation papers and was shown to yield stable and good performance. For the first 20 epochs, the learning rate is set to 0.001 and then decreased exponentially for fine-tuning with: learning rate $= 0.001 \cdot e^{0.05(20-\text{epoch})}$. Adam is used to apply the gradient to the model [49]. Experiments with normal SGD without momentum did not converge in any reasonable time. A batch size of 8 data points per batch is used, if not specified otherwise. The Glorot Uniform distribution is taken to initialize all model parameters [93]. The training is calculated on two NVIDIA GTX 1080Ti in parallel.

The final separation will be achieved through the normal ratio mask (equation 2.30 with $\alpha = 2$), because the calculation of the MWF through EM is an expensive calculation. The best performing model will then be evaluated with the MWF (equation 2.31). This is a common practice in the evaluation process of DNN based source separation. The MWF is calculated with the python-based toolbox Norbert[1]. After masking the complex-valued mixture spectrogram, the final source signals are obtained by performing an inverse STFT over the obtained source spectrograms.

[1] `https://github.com/sigsep/norbert`

3.4. Baseline Model and Network Architecture

Since there is a large variety of already existing DNN architectures for the source separation problem, a subset of well-known architectures was chosen to serve as a baseline for this thesis. These baseline models yield the current State-of-the-Art results, each representing a different category of network architecture (i.e. RNN, CNN, etc.). They will serve as the final evaluation benchmark.

1. **Spleeter**: A fully convolutional U-Net, which is an adaptation of the original U-Net by Jansson et al. [77] implemented by researchers of the music streaming service Deezer. The model was trained on a large private database ($\approx 24,000$ songs). They also present results when only trained on MUSDB18 for a fair comparison [81].

2. **OpenUnmix**: A recurrent neural network architecture based on three LSTM layers. It was specifically created as an open-source model for other researchers. It achieves the comparably good evaluation results with a recurrent architecture, without any convolutional layers [84].

3. **MMDenseLSTM**: The best performing model from the SigSep 2018. It is based on a densely-connected convolutional architecture and processes the spectrum in multiple bands. It also utilizes a recurrent LSTM to improve the performance of its predecessor MMDenseNet. Because of it's densely-connected convolutional structure, it has the lowest amount of model parameters of all models mentioned here (1.3M parameters). It was developed by researches of Sony and trained with additional 800 songs. Evaluation results are also available when only trained on MusDB18 for a fair comparison. [54].

4. **Demucs**: The newest model of all considered benchmarks used for this evaluation. It was proposed by a research team of Facebook and performs separation in the time-domain, in contrast to all former models, which are based on the TF representation. It is also the only model using a multitasking approach, separating all sources with a single network. It is built in a 1D convolutive U-Net structure with an additional recurrent layer and is trained on 150 extra songs. Evaluation results are also available when only trained on MusDB18 for fair comparison [57].

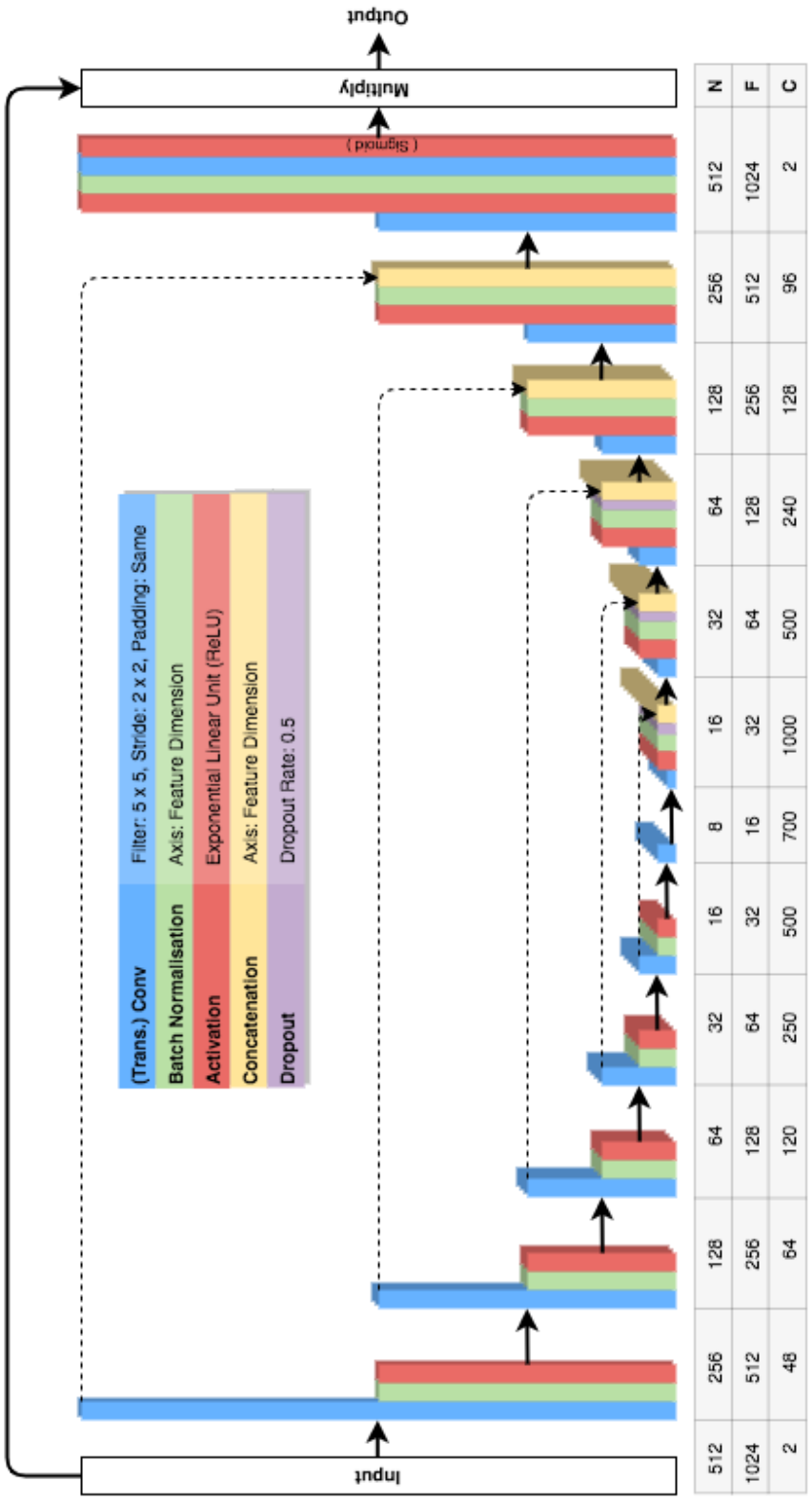

Figure 3.1.: Symbolic sketch of the original Spleeter baseline implementation. The color of each bar encodes the type of layer it represents. The height of the columns symbolises the size of the frequency (F) and time (N) dimension. The depth of the bars represents the amount of feature maps (C) it contains.

Initial experiments with both publicly available models, namely Open-Unmix and Spleeter, have shown a much faster convergence in terms of required epochs as well as much faster calculation time per epoch with the fully convolutional U-Net architecture of Spleeter. Therefore, the here proposed network architecture is based on Spleeter.

The public available implementation of Spleeter (from `https://github.com/deezer/spleeter/blob/master/spleeter/model/functions/unet.py`) is used and converted from Tensorflow 1.0 to 2.0. Figure 3.1 is a symbolic representation of the complete architecture. The model consists of six fully convolutional encoder and decoder stages with typical U-Net like skip-connections between each encoder/decoder pair. Each stage consists of a two dimensional strided (encoder side) or transposed (decoder side) convolution layer, a batch normalization, for faster and more stable convergence [78], and a non-linear activation function. In the last stage, an internal mask is created by using a sigmoid activation function in the last layer and multiplying it on the input tensor.

Since some of the design choices seem not optimal, the internal flow of information is restructured. The number of total layers is doubled by adding a transition layer (another convolutional layer without a down- or up-sampling operation) on every stage before re-sampling the feature maps. A max-pooling layer is used instead of a strided convolution as the down-sampling operation. These adaptations are inspired by the original U-Net design proposed in [76]. I assume that the transition layer and max-pooling operation improve the information flow inside the network, as it separates the filtering and the down-sampling process into two dedicated tasks. By replacing the strided convolutions with a max-pooling layer the amount of detail inside the feature maps is increased and selectively down-sampled instead of simply skipping every second value. A symbolic visualization of the architecture can be seen in figure 3.2. A detailed description of the full Spleeter architecture as well as the here proposed changes with information about the hyperparametrization of each layer (i.e. kernel size, number of feature maps, etc.) can be found in appendix A.

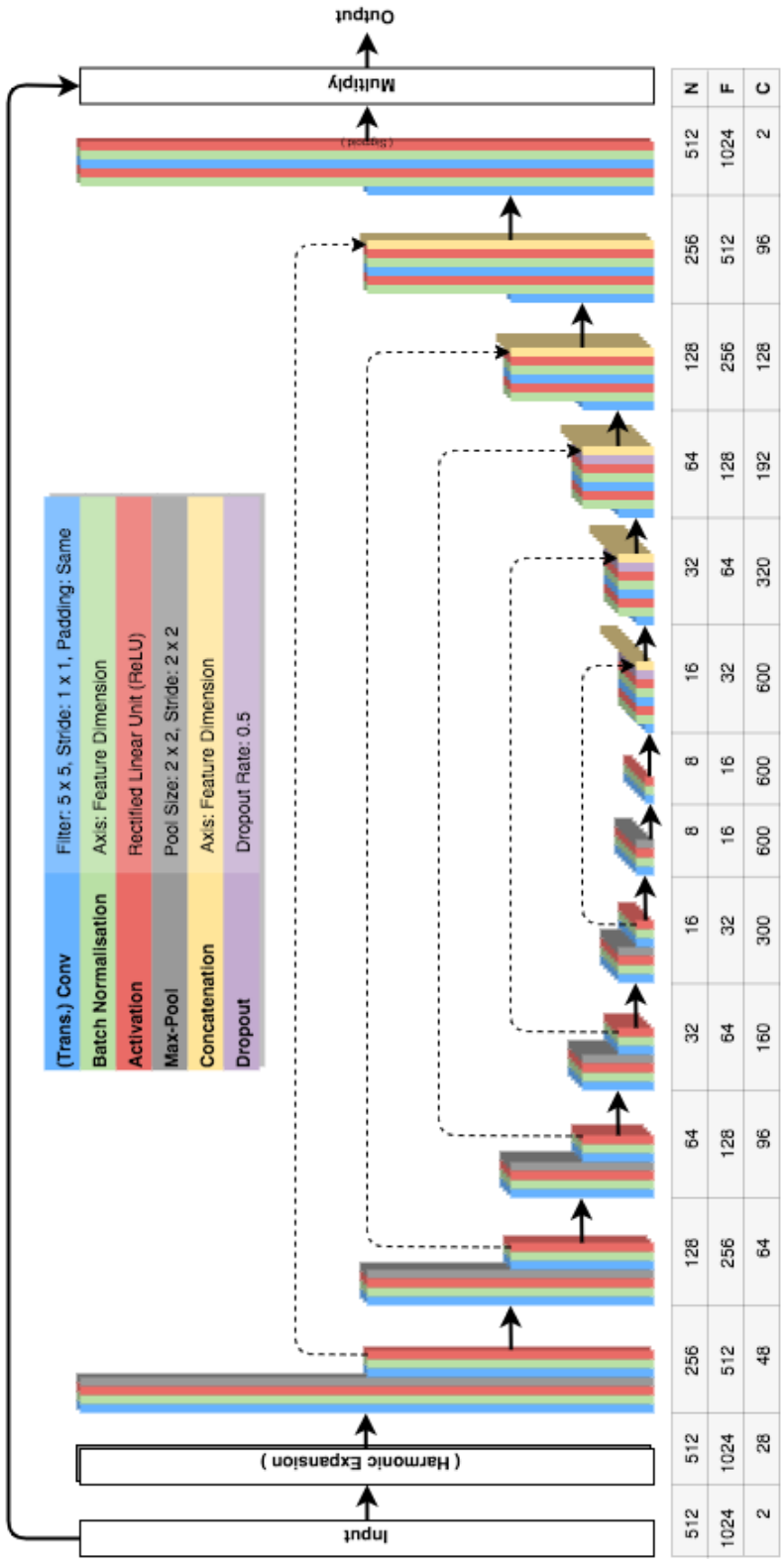

Figure 3.2.: Symbolic sketch of the proposed U-Net implementation. The color of each bar encodes the type of layer it represents. The height of the columns symbolises the size of the frequency (F) and time (N) dimension. The depth of the bars represents the amount of feature maps (C) it contains.

3.5. Model Extensions

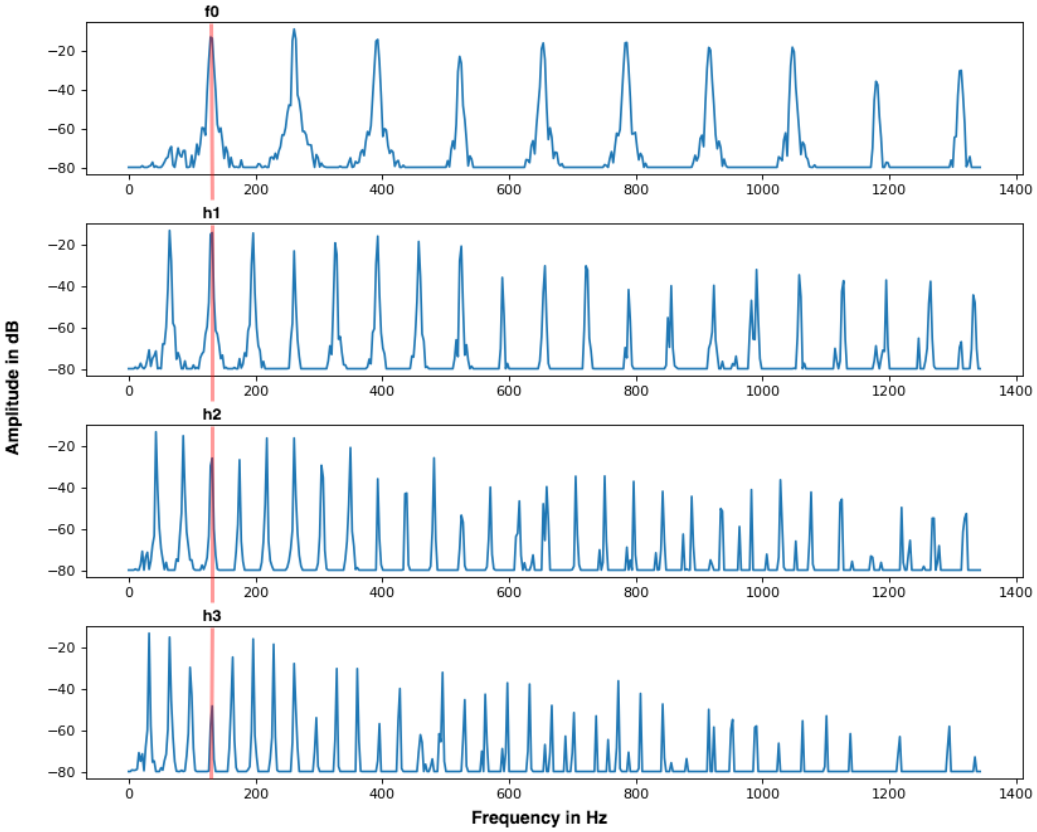

Figure 3.3.: Example of the proposed harmonic feature maps. The first row shows the original spectrum of a piano note, under that the first three aligned harmonic feature maps are shown.

Harmonic Expansion

Inspired by the issues of the non-linear scaled harmonic relationships in the frequency dimension, a novel technique is proposed to better utilize the harmonic structures in a CNN based network where a high amount of detail is required in the output. The method is based on the harmonic constant-Q representation proposed by Bittner et al. for CNN-based F_0 estimation [94]. They propose a three-dimensional input tensor $\mathbf{X} \in \mathbb{R}^{N \times F \times H}$, where N, F are the standard spectrogram dimensions and H is a harmonic dimension, listing the first H harmonic bins over the feature dimension of the input to the CNN. This harmonic expansion is adapted to the normal STFT domain. Therefore, an index matrix $\mathbf{G} \in \mathbb{Z}^{F \times H}_{[0,1,\ldots,F]}$ is constructed. Each element in $G[f, h]$ represents the index inside the spectrum, indexing the harmonic h of the fundamental frequency at index f. The

indices are simply calculated with: $G[f, h] = f \cdot h$. The left side of figure 3.4 shows the first eight entries of the harmonic index matrix with five harmonics. The matrix is then used to index each channel $i = 1, ..., I$ of the input spectrogram $\mathbf{X} \in \mathbb{R}_{\geq 0}^{N \times F \times I}$ to create the harmonic feature maps: $\hat{\mathbf{X}}_h[n, f, i] = \mathbf{X}[n, G[f, h], i]$. Figure 3.3 shows the original spectrum and the first three harmonic feature maps of a single spectrogram frame of a piano note. It can be seen that the harmonics of the fundamental frequency at around 150 Hz align perfectly over the feature map dimension (red line). This alignment is independent of the fundamental frequency and affects all frequency equally. The harmonic feature maps are then concatenated over the channel dimension to create the final harmonic aligned input tensor $\hat{\mathbf{X}} \in \mathbb{R}_{\geq 0}^{N \times F \times IH}$. If the index $G[f, h]$ exceeds the highest frequency bin F, the value in that feature map is simply set to 0.

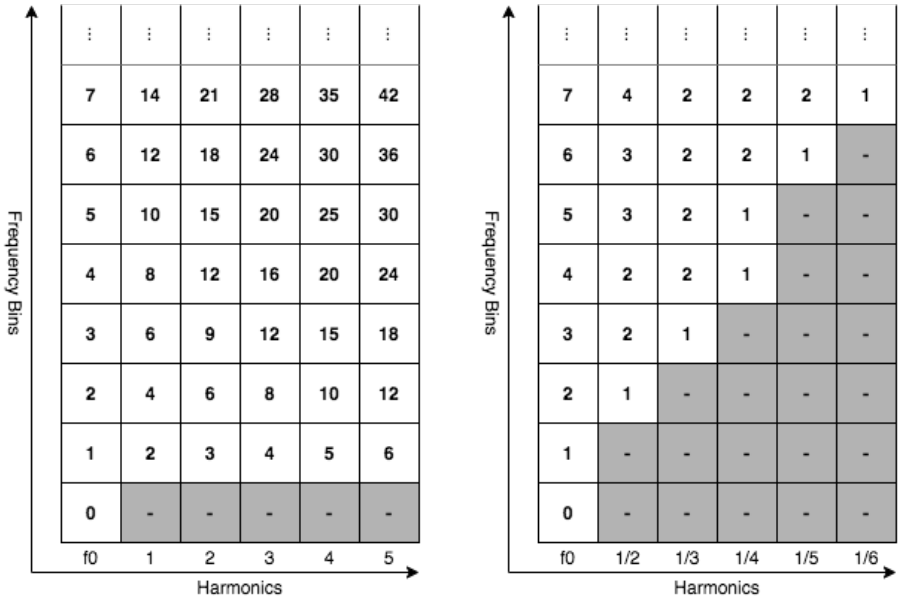

Figure 3.4.: Visualization of the first 8 entries of the harmonic index matrix. On the left, the first 5 harmonics are shown and on the right, the first 5 sub-harmonics are shown. The sub-harmonic indexes are rounded to the nearest integer.

Because a frequency bin can also be the harmonic of another lower frequency bin, sub-harmonic relationships are included as well in the harmonic feature maps. Therefore, rational fractions (e.g. $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}$,...) are also included in the harmonic index matrix. The sub-harmonic h is calculated by: $G_{sub}[f, h] = \frac{f}{h}$, which is then rounded to the nearest integer. If the index $G_{sub}[f, h]$ is lower than 1.0, the value in that feature map is set to 0. A representation of the sub-harmonic index matrix with the first five sub-harmonic can be seen on the right side of figure 3.4.

Furthermore, the sub-harmonic index matrix is extended, to also include other harmonics of that sub-harmonic above and below f by including all other rational fractions up to 2.0 (e.g. $\frac{1}{2}, \frac{3}{2}, \frac{1}{3}, \frac{2}{3}, \frac{4}{3}$,...). I call these dense harmonic feature map extensions. They are also concatenated to the final harmonic aligned input tensor. Hence, the final CNN has access at every frequency bin f to the harmonic series of f, to the related sub-harmonics of f as well as to their harmonic series around f.

The harmonic extension technique is implemented in a separate layer and used as the input to the proposed model, which can be seen in figure 3.2. To simplify the notation of the amount of sub-, dense-, and normal harmonics a shortcut notation is used referring with 'h' to the number of normal harmonics, 's' the number of sub-harmonics, and 'd' to the used dense harmonics. For example '5h+1s' refers to a harmonic extension including the first 5 harmonics and the first sub-harmonic in the harmonic feature maps. Logically the dense harmonic extensions always include the sub-harmonics as well.

Architectural Extensions

State of the Art source separation networks such as MMDENSELSTM [54], the dilated Conv-GRU Network [95], Demucs [96], and OpenUnmix [84] all benefit from additional recurrent layers inside their model architecture. Therefore, the proposed U-Net architecture is also expanded by an additional recurrent layer inside of the proposed U-Net structure to evaluate the effect of the recurrent extensions. As shown in [54] the implementation of the recurrent unit in deeper layers of the network performs better. Doing so, the network utilizes the prior convolutional layers to extract high-level features and uses the recurrent layer to model the dynamics of these features over time. Hence, the deepest transition layer inside the proposed U-Net is replaced by a Gated Recurrent Unit (GRU). To maintain a higher time resolution for the GRU only the frequency dimension is reduced in the deepest two layers, instead of reducing both frequency and time dimension. The frequency and feature dimensions are then flattened into a single dimension and a 1×1 convolution is used to reduce the size of this dimension to an appropriate dimension for the GRU. A bidirectional GRU unit is used and the output of both directions are concatenated and then re-scale through another 1×1 convolution back to the original dimensionality. This can be seen in the symbolic visualization of figure 3.5.

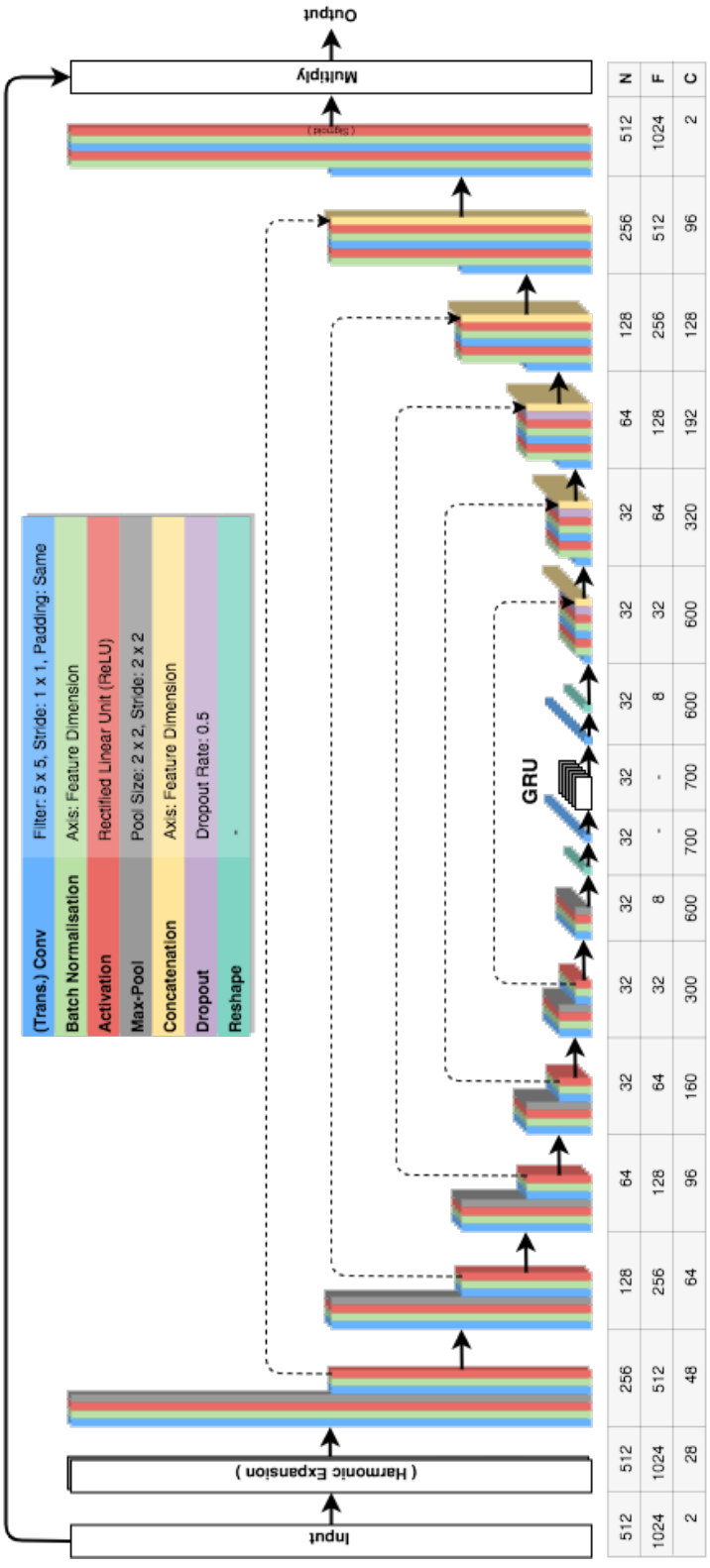

Figure 3.5.: Symbolic sketch of the GRU extension to the proposed U-Net. The color of each bar encodes the type of layer it represents. The height of the columns symbolises the size of the frequency (F) and time (N) dimension. The depth of the bars represents the amount of feature maps (C) it contains.

Another aspect, from which MMDENSELSTM yields its State of the Art results, is its densely-connected convolutional structure. Due to its ability to re-process feature maps at every layer, it is a very parameter efficient network structure [85]. To evaluate the effect of the densely-connected structure as well as the parameter efficiency of this structure, the proposed U-Net is expanded with a densely-connected encoder structure. Therefore, the whole encoder of the U-Net is replaced with a densely-connected encoder. The new encoder consists of three densely-connected blocks each followed by a down-sampling stage. Figure 3.6 shows the proposed densely-connected U-Net. One dense block consists of N densely connected layers, where each layer consists of a batch normalization followed by a ReLU non-linearity and a final two-dimensional convolution with M filters of size 5×5, as well as a concatenation layer, concatenating the output of all previous layer with the output of the current layer. This structure is an adaptation of the MMDENSENET encoder structure. The decoder block of the U-Net remains the same, to maintain a high resolution in the up-sampling path. To also maintain the same amount of skip-connections as in the normal U-Net, the down-sampling stages consist of two separate max-pooling operations with one skip-connection between them, which are divided by a convolutional layer used to reduce the feature map amount by a factor of θ after every dense block. The dense block and down-sampling block structures are also shown symbolically in figure 3.6. The used amount of kernels per Layer M, the number of layers per dense block N and feature map reduction factor θ for all dense blocks are shown in table A.2.

Multiple Networks vs. Single Network

To evaluate if the proposed U-Net performs well while separating multiple sources thru a single network, the M-U-Net structure from [83] is implemented in the proposed U-Net. The implementation is straightforward. The number of output channels in the last layer is simply changed from $I = 2$ to $J \times I = 8$ channels and then trained with all source magnitudes stacked as a single tensor $\mathbf{S} \in \mathbb{R}^{\hat{N} \times F \times JI}$. This effectively reduces overall utilized model parameters as well as the training time by a factor of $\frac{1}{J}$. Also their proposed weighting strategy 'EBW-P1' is used, with: $w_j = E_{max}/E_j$, which weights the gradient of each source according to the overall energy distribution, where E_j is the averaged energy of source j over all data points and E_{max} is the energy of the source with the highest energy. This strategy has shown better results in my initial experiments than the non-weighted training, as it avoids over-fitting of the model towards instruments with higher energy. The obtained weights for each source can be seen in table A.3. The results of the M-U-Net are compared with the results of the dedicated U-Nets to evaluate these different strategies.

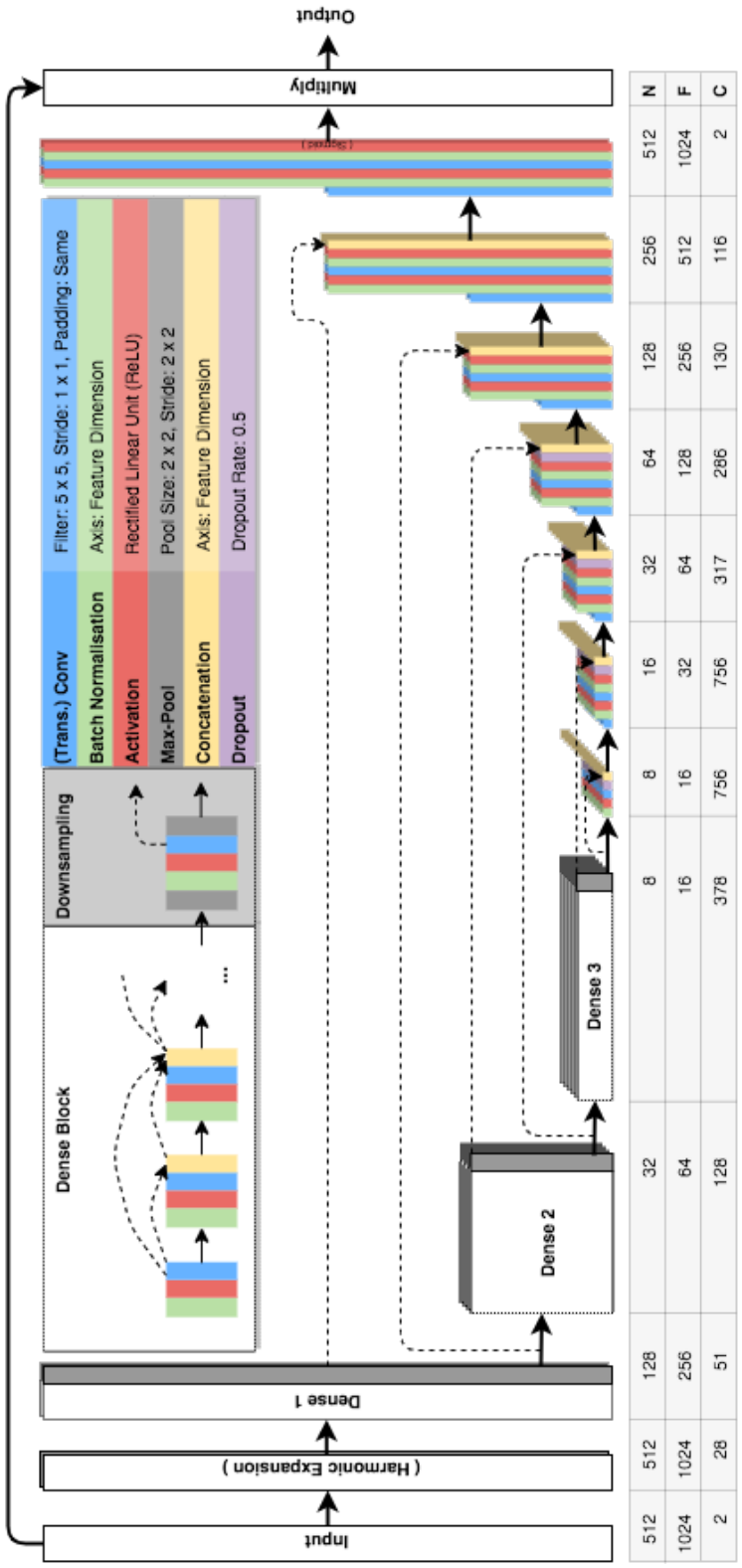

Figure 3.6.: Symbolic sketch of the densly-connected encoder extension in the proposed U-Net. The color of each bar encodes the type of layer it represents. The height of the columns symbolises the size of the frequency (F) and time (N) dimension. The depth of the bars represents the amount of feature maps (C) it contains.

3.6. Phase Alignment

Finally, an approach to solve the sound quality problems, caused by the 'noisy' mixture phase of the TF mask separation is proposed here. Since recent waveform-to-waveform models have been shown to extract the correct source phase information [57], an additional time-domain post-processing network is used to align the phase of the source signals separated by the main TF masking network. Defossez et al. have noted that Conv-Tasnet extracted sounds with hallow transients, similar to the one caused by the TF mask. Therefore, their proposed Demucs separation network, which also showed the highest sound quality in their perceptual evaluation study [57], is adapted for this task.

As the task of aligning the phase of an already separated source signal should be 'easier' than the whole source separation itself, the last two encoder/decoder stages together with the recurrent LSTM layer of Demucs are discarded to reduce the depth and computational effort of the network. Instead, only the first five encoder/decoder blocks are implemented. Demucs uses a combination of several layers per block. First, the input is processed by a convolutional layer with a large kernel as well as a large stride and the output is sent through a ReLU non-linearity. Then the feature maps are re-processed by a 1×1 convolutional layer and doubled in number, to be finally sent through a Gated Linear Unit (GLU), which acts as an internal feature map gate. The GLU was shown to be beneficial to the network performance [57]. The here proposed Phase Alignment Network (PA-Net) uses the same encoder structure as Demucs, with the exception that a batch normalization layer is used after each convolution, for a faster convergence, which was not used in the original Demucs implementation[1]. The decoder blocks are modified further: Instead of using a transposed convolutional layer with a large stride for up-sampling, a combination of linear interpolated up-sampling and 1D convolutional layers with kernel sizes of 3 is used. This was done to avoid aliasing in the up-sampling path, similar to the design of Wave-U-net [87]. The decoder blocks are implemented with additional batch normalization layers as well.

The input of PA-Net is set to 34304 samples, yielding a maximum receptive field of only 2 seconds. Since the whole separated source tracks are already available, it could be beneficial for the PA-Net to have access to more than 2 seconds of it. But instead of simply increasing the input size, a different strategy to achieve a wider context field is proposed here. Music is in general repetitive and the network could use these repetitive phrases to average out the artefacts and residual noises, introduced during the previous separation process. Hence, a novel attention mechanism is proposed, to efficiently enlarge the receptive field of the network to the relevant repetitive information of the full source track. The idea is inspired by the similarity-based median filtering approach by Rafi et al. [36], where they use the correlation between different magnitude frames of the mixture to find repeating sequences in a song.

[1] They use a different gradient weighting strategy instead.

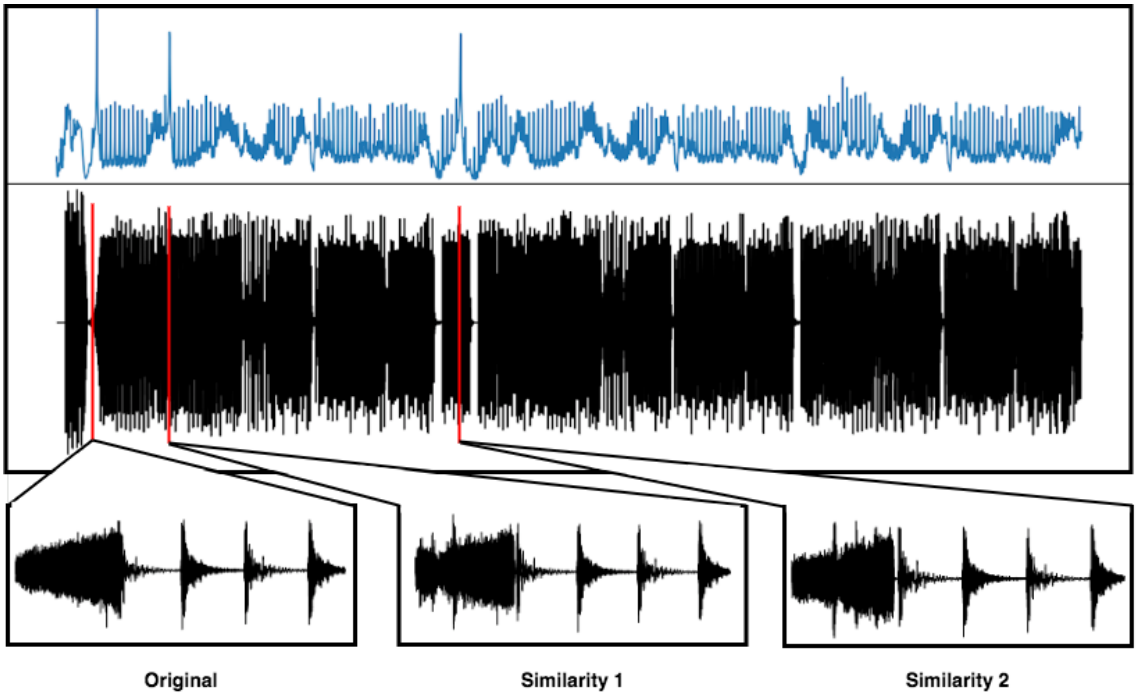

Figure 3.7.: Example of the proposed similarity measure, plotted for one specific frame block of a drum track. The red lines in the waveform mark the discovered repetitive phrases.

Similarly, the estimated source spectrograms of the separation networks are used to find repetitive phrases inside each source track. But instead of simply calculating the similarity matrix for each estimated source spectrogram $\hat{\mathbf{S}}_j$ (i.e. $\mathbf{G}_j = \hat{\mathbf{S}}_j \hat{\mathbf{S}}_j^T$), also referred to as the Gram matrix, a cosine-similarity inspired correlation is calculated over blocks of several magnitude frames of length L:

$$g_{j,m,\hat{m}} = \sum_{i=1}^{I} \frac{\sum_{n,f} (\widetilde{\mathbf{S}}_{j,m,i} \odot \widetilde{\mathbf{S}}_{j,\hat{m},i})[n,f]}{||\widetilde{\mathbf{S}}_{j,m,i}||_F \cdot ||\widetilde{\mathbf{S}}_{j,\hat{m},i}||_F} \tag{3.2}$$

Here, $\widetilde{\mathbf{S}}_{j,\hat{m},i} \in \mathbb{R}^{L \times F \times I}$ is one block of magnitude frames starting at frame m and ending at frame $m + L$. The $||\cdot||_F$ operator denotes the Frobenius norm. Hence, $g_{j,m,\hat{m}}$ describes the similarity between the magnitude block starting at m and the magnitude block starting at $\hat{m}$. It is calculated for every pair $[m, \hat{m}]$ in the estimated source spectrogram. Afterwards, a peak finding algorithm is used to find the most similar block pairs. Figure 3.7 shows an example of a similarity curve of one specific block, taken from a drum track. It can be seen that the peaks in the similarity curve (blue curve on top) correspond to very similar phrases in the source track. Especially for the source classes drums and bass in MusDB18, this method was able to find pairs of almost identical phrases.

The Phase Alignment Network input size of 2 seconds (corresponding to magnitude blocks of $L = 64$ frames) was chosen, since this yields approximately 1-2 bars of music, depending on the tempo of the song. This was considered as a good ratio to yield repeating sequences of a song, which are almost identical. The original phrase and its K most similar phrases, detected by the peak finding algorithm, are stacked over the channel dimension into one tensor. Initially this tensor was simply provided as the input of the encoder. However, the network was not able to gain any information from this representation. Although the phrases contain a very similar 'musical' content, their waveform representation is highly detailed and the phrases vary too much in this representation. Therefore, the final PA-Net processes each phrase separately through the same encoder, yielding $K + 1$ encoded representations (i.e. one for each similarity phrase and one for the original phrase). To create a useful representation for the decoder from all these encodings, a simplified self-attention mechanism is used. The original self-attention mechanism uses a compatibility function, which is often a modified dot-product with a learned linear transformation, to calculate feature similarities between different elements of a sequence and uses another learned soft-max function to aggregate them [55]. Since the waveform encodings are still high dimensional, the learned linear transformations would add an unreasonable amount of parameters to the model and thus are not implemented. Instead, only the compatibility function and the soft-max function are kept without any learned transformations. In the implemented self-attention layer the correlation between every encoded similarity phrase $\mathbf{h}_k$ and the encoded original phrase $\mathbf{h}_{orig}$ is calculated with a simple dot-product:

$$e_k = \mathbf{h}_k^T \, \mathbf{h}_{orig} \quad \text{for every } k = 1, 2, ..., K \tag{3.3}$$

The final attention embedding $\mathbf{h}_a$ is created by summing up all encoded similarity phrase using a soft-max weighting:

$$\mathbf{h}_a = \sum_{k=1}^{K} \alpha_k \cdot \mathbf{h}_k \quad \text{with } \alpha_k = \frac{exp(e_k)}{\sum_{i=1}^{K} exp(e_i)} \tag{3.4}$$

The attention embedding includes all the relevant information of the K encoded similarity phrases, weighted according to their correlation to the encoded original. Since this attention mechanism does not include any extra parameters or heavy computational effort, it is implemented after every encoding stage of the PA-Net. The attention embedding $\mathbf{h}_a$ are concatenated with the original encoded states $\mathbf{h}_{orig}$ over the feature dimension and form the skip-connections between each encoder and decoder pair. The complete proposed method together with the PA-Net architecture is shown in figure 3.8.

The PA-Net was trained with the source estimates of the proposed U-Net, separated through MWF as the input and the original source tracks as the target. The L1-Norm over the difference between estimate and target was used as a loss function since it is phase-sensitive and showed good results in the original Demucs network [57]. One PA-Net with 15 million parameters was trained for every source. The network is trained

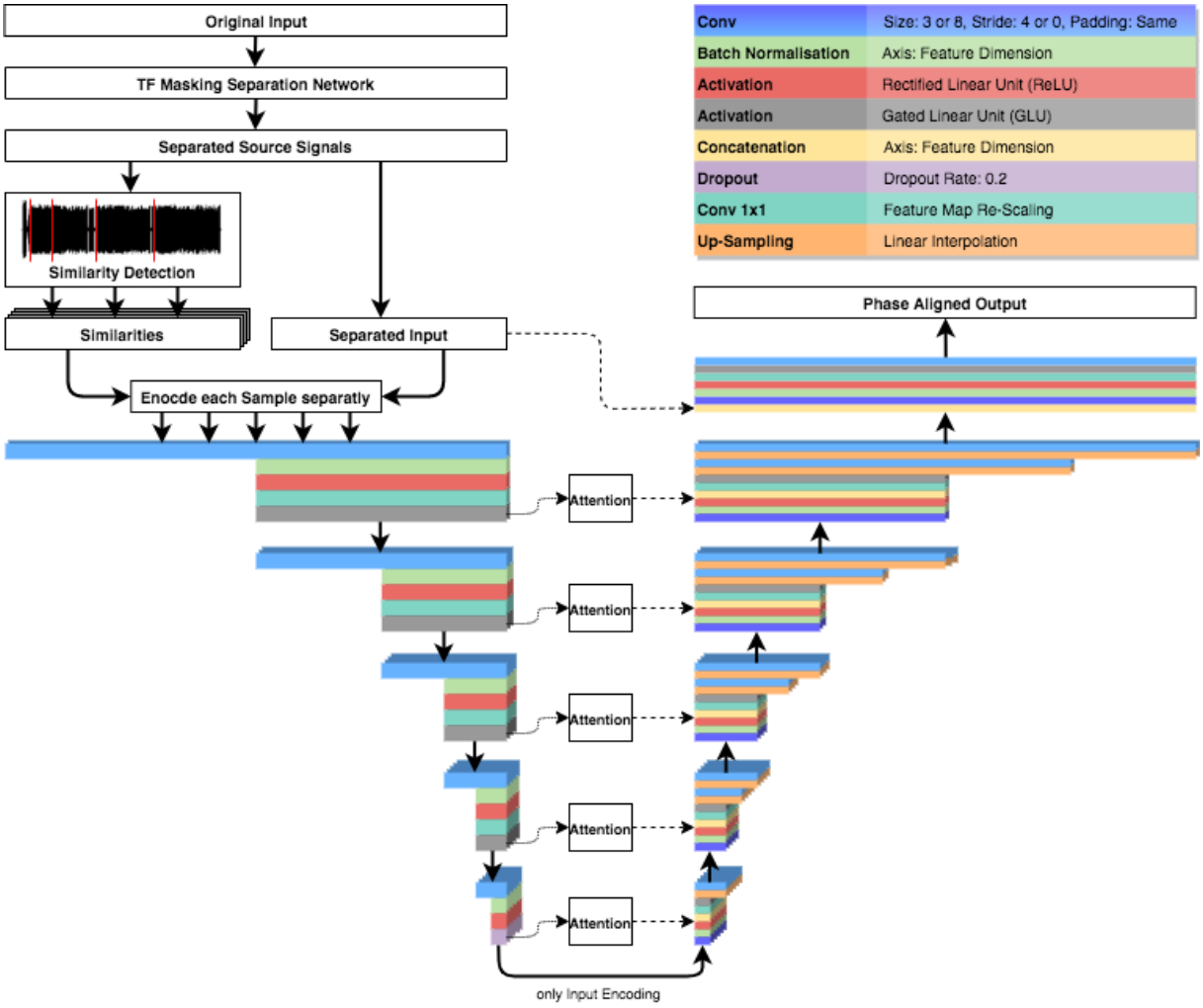

Figure 3.8.: Symbolic sketch of the Phase Alignment Network, with the Similarity Attention Mechanism. The color of each bar encodes the type of layer it represents.

under very similar settings, as described in chapter 3.3, using a learning rate of 0.001 with an exponential decay over 60 epochs. Since the data points are much shorter, 96 examples were taken per song, instead of only 32. Seven similar phrases ($K = 7$) are detected with the previously described method and given together with the original phrase to the network. To avoid clicks between two consecutive blocks of 2 seconds aligned by the PA-Net, an overlap-and-add procedure was used, with a hop-size of 1 second (i.e. overlap of 50%) together with a Hann-window to create the final phase-aligned source track.

4. Evaluation

4.1. Evaluation Metrics

Finding good numerical performance measures to track the separation quality of different audio source separation models is an important issue that made the comparison between different separation methods hard and also subjective. To tackle this issue Vincent et al. proposed a systematic evaluation method in [97] and made it publicly available as a MATLAB toolbox[1]. The toolbox has become the evaluation standard and is used in hundreds of papers as well as in the Signal Separation Evaluation Campaign (SiSep) [98]. They also released a reference implementation in Python, the MusEval toolbox[2]. Their proposed evaluation metric is inspired by the classical Signal-to-Noise Ratio (SNR), given by:

$$\text{SNR} = 10\ log_{10}(\frac{p_s}{p_n})\ \text{dB} \tag{4.1}$$

Where p_s and p_n are the power of the signal and noise respectively. In the source separation context the definition of p_n, the energy of the unwanted components which are not part of the source signal, is not trivial. Therefore Vincent et al. proposed to divide the estimated source signal energy $p_{\hat{s}}$ into four parts:

$$p_{\hat{s}} = p_s + e_{inter} + e_{arti} + e_{noise/spat} \tag{4.2}$$

Where p_s is the contained power of the original source signal inside the source estimate, e_{inter} is the power of the interfering other source signals, e_{arti} is the power of artifacts, which emerge during the separation process, and e_{noise} is the power of additional sensor noise [97]. Because e_{noise} requires complete knowledge over the sensor noise signal, which is mostly unknown, it got discarded in the second version of the toolbox and instead replaced by $e_{spat.}$, which measures the spatial distortion of the estimated source image [99]. The decomposition is computed through several projections, which can be found in detail in [97] and for the multi-channel version in [99].
The decomposition is then used to calculate energy ratios to evaluate the relative amount of each of these four terms. The main evaluation metric is the Signal to Distortion Ration (SDR), which measures the log-ratio between p_s and the overall error energy:

$$\text{SDR} \ = \ 10\ log_{10}(\frac{p_s}{e_{inter} + e_{arti} + e_{noise/spat}})\ \text{dB} \ = \ 10\ log_{10}(\frac{||s||^2}{||s - \hat{s}||^2})\ \text{dB} \tag{4.3}$$

[1] `http://bass-db.gforge.inria.fr/bss_eval/`
[2] `https://github.com/sigsep/sigsep-mus-eval`

Besides the SDR, which incorporates all possible kinds of distortion arising from different source separation algorithms, they proposed three other ratios for a more detailed evaluation:

1. The Signal to Interference Ratio (SIR) accounts for the interference of other unwanted source signals inside the estimate:

$$\text{SIR} = 10\ log_{10}(\frac{p_s}{e_{inter}})\ \text{dB} \tag{4.4}$$

2. The Signal to Artifacts Ratio (SAR) indicates the amount of introduced distortion, for example, 'musical' noise and 'gurgling' artifacts in the source estimate:

$$\text{SAR} = 10\ log_{10}(\frac{p_s}{e_{arti}})\ \text{dB} \tag{4.5}$$

3. The Image to Spatial distortion Ratio (ISR) accounts for the spatial distortion in the multi-channel case (e.g. stereo image):

$$\text{ISR} = 10\ log_{10}(\frac{p_s}{e_{spat}})\ \text{dB} \tag{4.6}$$

Because the decomposed power-terms, as well as the perceived separation quality varies across time, the calculation of these ratios is usually done locally on windowed frames of the signal and then averaged. A rectangle window is sufficient [97].

As already mentioned, there is a public reference implementation in Python, which is used by most researchers for the evaluation of their model. For this thesis, I also use this implementation. During initial experiments, a change of up to $\pm 0.5\,\text{dB}$ in the metrics was noted depending on the evaluation hyperparameter. Since these hyperparameters are rarely mentioned, the initial values given by the toolbox were used. The frame and hop size are both set to one second and a rectangular window is used. The given validation and test set splits of the MusDB18 dataset are re-sampled to 16 kHz and used separately for evaluation. The final evaluation yields source and time-dependent metrics, respectively for all songs in the validation and test set:

$$\text{SDR}_{n,j},\ \text{SIR}_{n,j},\ \text{SAR}_{n,j},\ \text{ISR}_{n,j}.$$

Figure 4.1 shows a typical density plot of $\text{SDR}_{n,j}$ taken over all sources, all frames, and all songs, of the test set. The distribution is heavy-tailed towards the left with many outliers towards negative values, as well as a spike at 0dB. Therefore, the mean is not describing the overall performance well, as can be seen, marked in green. It is a very commonly used practice to use the median of the distribution instead, shown in orange. The SIR, SAR, and ISR follow a very similar heavy-tailed distribution.

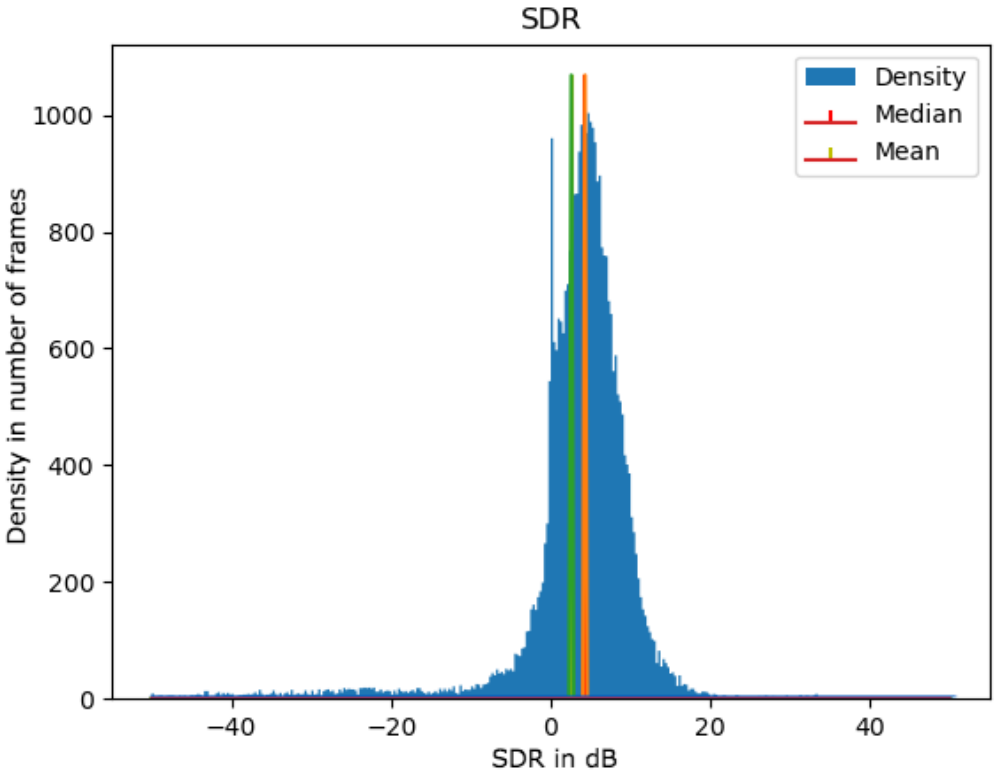

Figure 4.1.: An example of the separation performance. The plot shows the density of the SDR distribution taken over all sources, all songs and all frames. The green bar indicates the mean, the orange bar indicates the median.

For the evaluation of this thesis the median, mean and standard deviation are calculated from the evaluation data in the following ways:

1. The statistics are calculated over all frames of a song. Afterwards, the statistics over all songs per source are calculated from the previously calculated statistics. This is the standard procedure given in the SiSep campaign. It prevents that longer songs have a higher weight in the evaluation.

2. The statistics are also calculated over all frames and all songs concatenated together as this has a more accurate resolution, especially in the median and standard deviation (although it is biased towards longer songs).

3. The same procedure as in 1. is applied but now all values from all sources are concatenated together before calculating the statistics to get an overall performance measurement.

4. The same procedure as in 2. with all values from all sources concatenated, is applied to get detailed (but biased) overall performance measurement. Figure 3.6 is created this way. The spike at 0 dB is only visible here and averaged out with the standard SiSep procedure (list item 1. and 3.).

Besides these different ways of calculating the statistics and the previously mentioned unknown evaluation hyperparameter settings, there is another well-known issue with this evaluation metric, particularly with the MusEval toolbox. The toolbox automatically calculates the optimal gain coefficient, time lag, source permutation, and an FIR filter with 256 taps between the estimates and the original sources [97]. As the evaluation toolbox was originally developed for general blind source separation *"one potential justification for this is that a reference may be available for a source signal instead of the spatial image at the microphone which recorded the noisy mixture, and that spatial image is likely to be close to the result of the convolution of the source signal with a short FIR filter, as an approximation to its convolution with the actual room impulse response (RIR). This however leads to a major problem, because the space of signals achievable by convolving the source signal with any short FIR filter is extremely large and includes perceptually widely different signals from the spatial image.* (in [100], page 627) For music source separation of professional studio productions, these FIR filters are completely unnecessary and can distort the whole evaluation results. This was shown under extreme conditions in [100], where Le Roux et al. filtered out almost all frequency components of a source but the SDR metric still showed a very good separation quality. They proposed the Scale Invariant Signal to Distortion Ration (SI-SDR), which is only invariant to a different gain factor between the estimate and the original source track. The SI-SDR is given by:

$$\text{SI-SDR} \;=\; 10\ log_{10}(\frac{||\alpha s||^2}{||\alpha s - \hat{s}||^2})\ \text{dB} \tag{4.7}$$

With the optimal gain coefficient $\alpha = \frac{\hat{s}^T s}{||s||^2}$ [100].

For the evaluation of all experiments performed in this thesis, the MusEval toolbox implementation of the SigSep is used on the test split of MusDB18 to make the results comparable to the results of other researchers. Here, the statistics of the evaluation data as described in the list items 1. and 3. are used. To detect distorted evaluation results in these metrics due to the FIR filter problem an informal listening test on the source estimates after each experiment is done. Furthermore, a version of the proposed SI-SDR metric as described in equation 4.7 is implemented in Python. As they propose the metric for single-channel evaluation, in the implemented version the left and right channels of the estimate and the original sources are simply summed up and the metric is calculated over the summed signals. It was noticed that the spike at 0 dB in figure 4.1, which is also prominent in the SIR, SAR, and ISR, does not exist in the SI-SDR implementation, as can be seen in figure 4.2, which shows the distribution of the SI-SDR of the same model.

This spike could be caused by a small normalization constant, which might be used for numerical stability in the MusEval toolbox. Compared to the SDR plot from figure 4.1, the SI-SDR distribution shows a much smoother distribution with a heavier left-sided tail. This might suggests that the MusEval SDR implementation covers up outliers with the FIR filter.

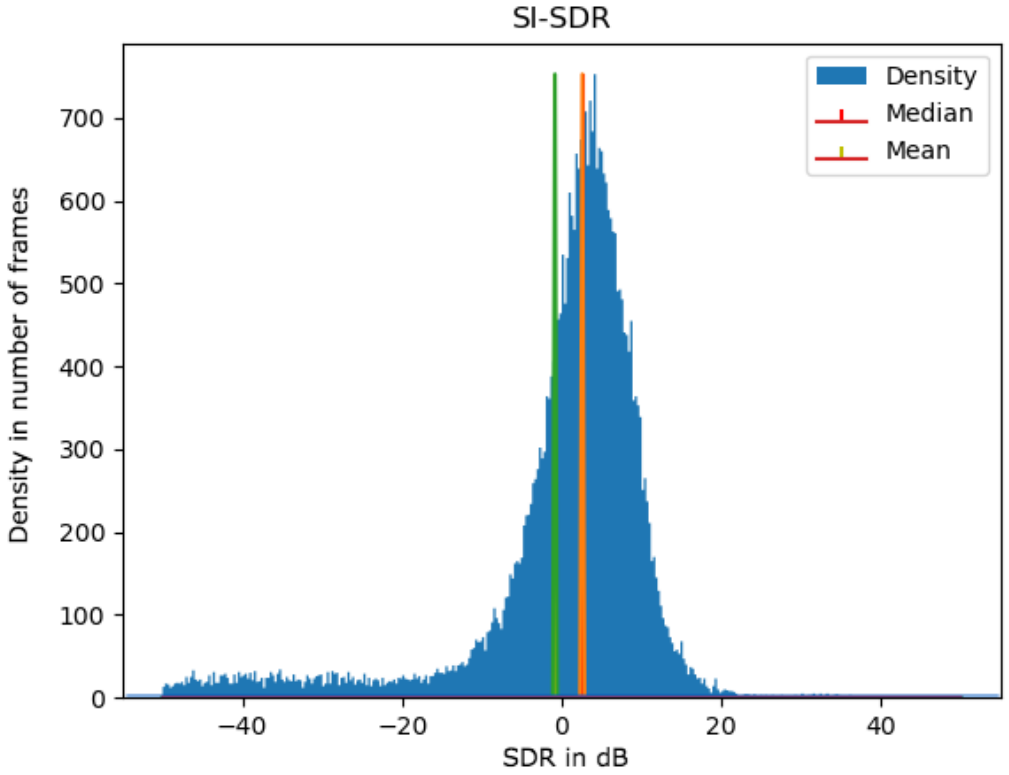

Figure 4.2.: The plot shows the density of the SI-SDR distribution taken over all sources, all test songs and all frames. The green bar indicates the mean, the orange bar indicates the median.

Besides the evaluation on the test split (50 songs), also the validation split (14 songs) is evaluated for every experiment. The complete statistics for all evaluation results (calculated as described in list 1 and 3) can be found attached separately to this thesis. During the evaluation in this chapter, only the median of the SDR and SI-SDR, taken over the test set, will be shown since it is much easier to grasp improvements.

As the values of these metrics might seem quite arbitrary, two oracle evaluation results are presented here. Firstly, the Ideal Ration Mask (IRM) is evaluated. The IRM is created by using the original source spectrogram as the estimate to create the ratio mask (eq. 2.30 with $\alpha = 2$). It is the maximum benchmark which can be achieved with a TF mask [14]. Note that this is not the universal maximum performance since time-domain methods can theoretically achieve a performance of $+\inf$ dB (if $s = \hat{s}$ in eq. 4.3). Secondly, the performance is evaluated when the mixture is taken as the estimate (Mix). This can be seen as a minimum benchmark, although again lower values are definitely possible, although any model performing worse than this benchmark would be pointless. Both benchmarks represents a good range to interpret the evaluation results. The benchmarks are shown in table 4.1. The complete benchmark results can be found attached to this thesis. Note that in the Mix benchmark, the ISR and SAR are not meaningful, since there are no spatial distortions or artifacts inside mixtures and these metrics theoretically should be at $+\inf$ dB.

	All		Drums		Bass		Other		Vocals	
	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR
IRM	9.09	9.03	9.76	9.26	7.97	8.13	8.37	8.83	10.47	10.31
Mix	−5.32	−5.57	−3.96	−4.43	−5.58	−6.71	−5.31	−5.15	−5.37	−5.43

Table 4.1.: Upper (IRM) and lowest (Mix) benchmark performance on the MusDB18 testset given in SI-SDR (SI.) and SDR.

4.2. Results

In this section, the evaluation results of the different DNN architectures, described in chapter 3, are shown and the best performing model is compared to the current State-of-the-Art models (listed in section 3.4). Since most of the papers do not contain the previously described metrics the final comparison is only done with the standard MusEval toolbox metrics.

In total seven different U-Nets were trained. If not mentioned otherwise, all models are trained under the conditions described in section 3.3. For a fair comparison the amount of feature maps is adjusted in each architecture, so all roughly have the same number of trainable parameters.[1] The number of trainable parameters is fixed to ≈ 30 million, which showed good results during initial experiments without indicating signs of over-fitting in the validation loss. The list below shows the different U-Nets with their respective abbreviation:

1. **S**: Original Spleeter architecture converted to Tensorflow 2.0 and trained with the previously described settings.
2. **U**: Proposed deeper U-Net architecture.
3. $\mathbf{U}_H$: Proposed U-Net architecture with additional harmonic extensions (5h+3d).
4. $\mathbf{U}_{H+A}$: Proposed U-Net architecture with additional harmonic extensions (5h+3d) and trained with augmented data.
5. $\mathbf{GRU}_{H+A}$: Proposed U-Net + GRU architecture with additional harmonic extensions (5h+3d) and trained with augmented data.
6. $\mathbf{Dense}_{H+A}$: Proposed densely-connected encoder U-Net architecture with additional harmonic extensions (5h+3d) and trained with augmented data.
7. $\mathbf{M\text{-}U}_{H+A}$: Proposed Multitask U-Net Architecture with additional harmonic extensions (5h+3d) and trained with augmented data

Note that the results should be interpreted with caution because this thesis cannot provide an empirical study with multiple training runs per model due to the computational complexity. One training with all instruments and including the evaluation took between 10 and 96 hours on two NVIDIA GTX 1080 TI depending on the model type. To provide at least some reliability, the U_{H+A} model was trained five times and the standard deviations of all results were calculated. The results show a deviation of $\approx \pm 0.1\,\mathrm{dB}$ per instrument due to randomly initialized parameters and the non-deterministic data pipeline. The metrics taken over all sources jointly (list item 3.) only varied with a standard deviation of $\approx \pm 0.05\,\mathrm{dB}$ and therefore can be seen as a quite stable indicator for the general model performance.[2]

[1] The amount of feature maps in each model are shown in appendix A.1.
[2] All standard deviations of the SI-SDR and SDR can be found in appendix B.1.

	All		Drums		Bass		Other		Vocals	
	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR
S	2.45	3.94	3.58	4.56	2.13	3.51	1.66	3.17	2.53	4.39
U	2.96	4.29	4.04	4.65	2.16	3.61	2.17	3.31	3.91	5.13
U_H	3.35	4.59	4.34	5.03	2.76	4.12	2.38	3.72	4.73	5.40
U_{H+}	**4.27**	**5.15**	**5.29**	5.56	**3.50**	**4.54**	2.73	4.12	5.71	6.03
GRU_{H+}	4.26	5.10	5.25	5.55	3.32	4.53	**2.84**	4.16	**5.82**	**6.17**
$Dense_{H+}$	4.06	4.98	5.28	5.47	3.14	4.53	2.75	**4.26**	5.68	6.08
M-U_{H+}	3.69	4.82	4.95	**5.60**	3.33	4.10	2.22	3.92	4.80	5.50

Table 4.2.: Median values of SI-SDR and SDR of the evaluation of different proposed separation network architectures.

Table 4.2 shows the median SI-SDR and SDR values of the main experiments trained under standard conditions, as described in section 3.3. The best results are highlighted in bold. At first, it can be seen that the proposed deeper U-Net outperforms the original U-Net in all cases (S vs. U), validating the assumptions made for the deeper U-Net architecture. Only in the case of the bass separation task, the results are too close, so they cannot be judged reliable. Next, the harmonic feature map extensions show a constant improvement in all instruments, outperforming the model without harmonic extensions throughout (U vs. U_H). The highest impact on the separation performance was caused by the data augmentation (U_H vs. U_{H+}). The additional augmented data improved the overall SI-SDR by almost 1 dB. The proposed U-Net with harmonic extended feature maps and augmented data (U_{H+}) shows the overall best performance. The evaluation results of both architectural extensions, the GRU and the densely-connected encoder, did not show any significant improvements over the normal U-Net architecture. Both perform minimally better in the categories other and vocals and perform worse in category bass. In general, they show a very similar performance to the normal U_{H+} model. Finally, the multitasking U-Net (M-U_{H+}), which separates all sources combined thru a single U-Net, performs worse than the source dedicated U-Nets (M-U_{H+} vs. U_{H+}).

Since the U_{H+} model showed the best evaluation results as well as a relatively fast training time (compared to GRU_{H+} and $Dense_{H+}$) it was chosen for the final benchmark comparison. Therefore the training settings were modified. Firstly, the validation set was included in the training set, adding 14 extra songs to the training set. This is a common practice and all other State-of-the-Art results shown here were obtained on the full training set of 100 songs. As this eliminates the validation loss during training, the model at the last epoch was evaluated instead of the model with the best validation loss. The number of data points per epoch was doubled from 3200 to 6400 and the batch size was reduced to 4, to increase the total amount of gradient updates. After 60 epochs the learning rate was decreased to 0.0001 and trained for another 30 epochs for the final fine-tuning of the model. In total each individual network has seen 576,000 data points and received 128,000 gradient updates. The training took around 20 hours per source

instrument. The final separation was then performed with both the ratio mask and the MWF, as described in chapter 3. Both evaluation results are shown in table 4.3. It can be seen that the additional data, as well as the longer training yields an improved separation performance in comparison to the models trained under the standard settings. The MWF improves the separation performance furthermore, as expected. Table 4.3 also includes the evaluation results of the full proposed separation method, using the proposed U-Net for MWF separation and the final PA-Net as a post-processor. The PA-Net was trained on the full training split separated with U_{H+}, GRU_{H+}, and the final MWF model, to not exclusively train on the residual errors of a single model. The PA-Net improves the separation performance particularly on the drums and bass, which was expected, as it was designed to alleviate the 'noisy' phase issue occurring in these source classes.

	All		Drums		Bass		Other		Vocals	
	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR
Mask	4.60	5.47	5.84	6.24	3.71	4.83	3.36	4.48	6.43	6.61
MWF	4.84	5.78	6.48	6.64	4.22	5.33	**3.49**	**4.78**	6.74	6.85
PA-Net	**5.08**	**5.83**	**7.08**	**6.96**	**4.47**	**5.63**	3.40	**4.78**	**7.03**	**6.88**

Table 4.3.: Median of SI-SDR and SDR of the final separation models, evaluated with both Ratio Mask and Multi-channel Wiener Filter (MWF) and send through the final Phase Alignment network (PA).

Table 4.4 shows the comparison to the current State-of-the-Art separation models. For completeness, the first block shows the results of the models that were trained with additional data. The second block shows the results which were only achieved with the MusDB18 without additional data. Here, the proposed method outperforms all current State of the Art models, except Demucs in the bass category. Especially in the category other, the proposed model performs much better, almost outperforming the currently best results achieved by MMDENSELSTM trained with additional 800 songs.

Model	Extra Data	Avg.	Drums	Bass	Other	Vocal
Demucs	150	**6.33**	**7.08**	**6.70**	4.47	7.05
MMDENSELSTM	804	6.04	6.81	5.40	**4.80**	**7.16**
Spleeter	24097	5.75	6.71	5.51	4.55	6.86
Proposed Method	0	**6.04**	**6.96**	5.54	**4.78**	**6.88**
Demucs	0	5.58	6.08	**5.83**	4.12	6.29
MMDENSELSTM	0	5.59	6.43	5.16	4.15	6.60
OpenUnmix	0	5.33	5.73	5.23	4.02	6.32
Spleeter	0	4.43	5.15	4.27	3.21	5.10

Table 4.4.: Benchmark comparison of the State-of-the-Art separation models compared to the proposed method evaluated on the test split of the MusDB18.

4.3. Ablation Study

This section provides a more in-depth evaluation of the achieved results and obtained hyperparameter settings. It also covers some interesting aspects, noted during the design of the final separation process.

Other changes in the U-Net architecture than the once already described in section 3.4 did not show any positive effects. Changing the kernel size of the convolutional filters to 3×3 or to 7×7 both decreased the performance slightly ($\approx -0.2\,\text{dB}$ SI-SDR). Nevertheless, it was noted that in the case of lower amounts of total trainable parameter, the 3×3 kernels eventually outperform the 5×5 kernels, when both models are trained with the same number of parameters. This indicates that a higher amount of feature maps is more important for the separation than a bigger convolutional kernel size. Both, the additional transition layers and the max-pooling operations showed the biggest impact on the performance, each improving the results by $\approx 0.2\,\text{dB}$ SI-SDR.

The proposed deeper U-Net architecture also showed two negative effects in comparison to the original model (S vs. U). Firstly, the proposed U-Net takes almost twice as long during training as the original Spleeter model.[1] This is probably caused by the additional transition layers as well as the bigger non-strided convolutions. Secondly, the proposed model showed an unstable convergence of the validation loss during training. An example for this can be seen in the lower plot of figure 4.3 in the blue curve, shown for the vocal separation model (the other instrument categories showed a very similar behavior). The validation loss shows heavy unstructured fluctuations during the convergence. This is not the case when training the original Spleeter architecture, which can be seen in the upper plot. One reason for this effect could be the deeper structure of the proposed architecture, which makes the learning process more unstable.

Apart from improving the separation performance by $\approx 0.4\,\text{dB}$ SI-SDR, the harmonic feature map extensions showed another interesting property, counteracting the before mentioned unstable validation loss convergence. This can be seen in the red curves of figure 4.3. By including the harmonic feature maps, the validation error converges much faster and much more stable. This is especially advantageous for the proposed deeper U-Net, which showed a very unstable convergence before, but the same effect can also be seen with the original Spleeter model in a weaker form. Interestingly, the improved convergence of the validation loss is only caused by the sub-harmonic feature map extensions and not by the normal harmonic feature map extensions. This can be seen also in figure 4.3 in the orange curve, which shows the validation error of the proposed model trained with the first five harmonic feature maps (5h) without sub-harmonics feature maps. Even though it converges to a lower minimum than the model without the extensions, it still shows a similar unstable convergence behavior. The stabilization effect is already caused by including only the first sub-harmonic feature map ($h = \frac{1}{2}$).

[1] Original: ≈ 9 hours, Proposed: ≈ 16 hours

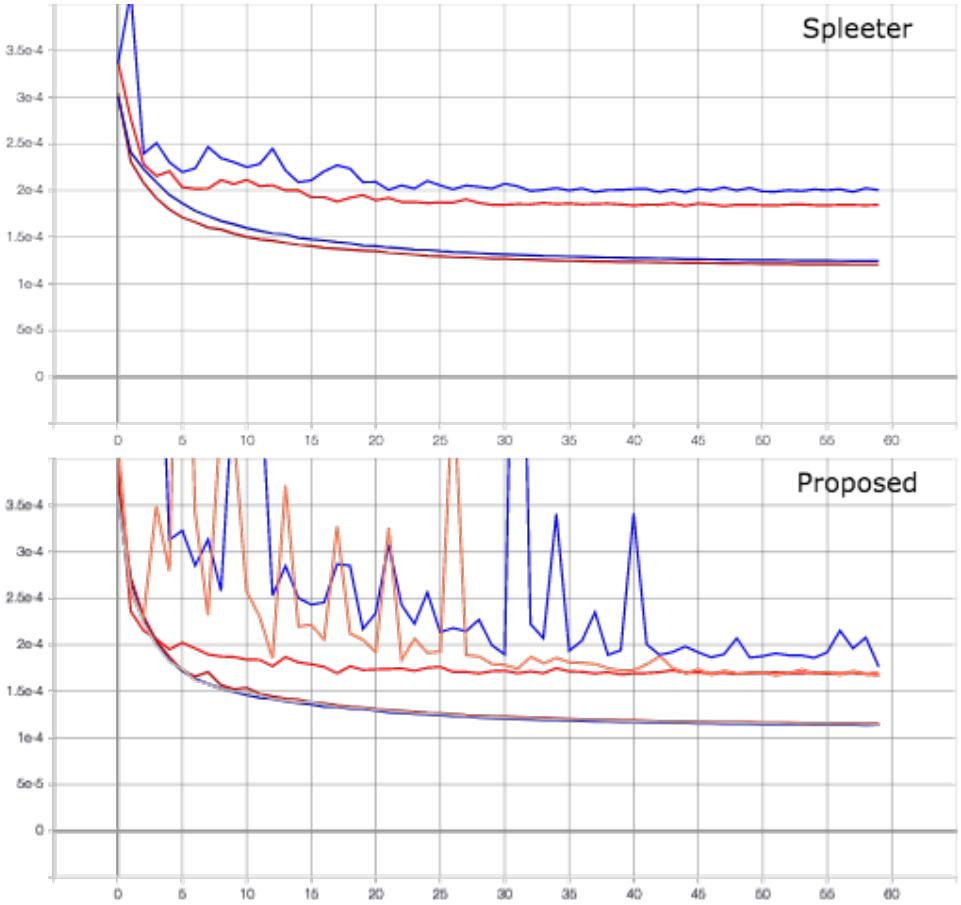

Figure 4.3.: Training- (dark) and Validation- (bright) Loss during vocal separation training. Blue: Model with no harmonic extensions, Red: Model with 5 harmonic and 3 subharmonics extensions, Orange: Model with 5 harmonic extensions. The x-axis shows all 60 epochs spaced equally and the y-axis shows the loss scaled linearly from 0-0.0004.

The dense harmonic extensions show the same improvements, as they automatically include the sub-harmonic feature maps.

A hyperparameter study of various harmonic feature map extensions was done and can be found in the appendix B.2. In general, all of the harmonic feature map extensions outperform the model without them in all metrics throughout, which confirms the assumptions that the harmonic feature maps help the CNN to learn the exponentially scaled harmonic relations. The effect was again verified by adding the harmonic feature map extensions to the original Spleeter model, which also yielded an improved performance. Analyzing the amount of included harmonics, it can be seen that more harmonic feature maps do not necessarily improve the results. Almost the opposite is the case.

The first three to five harmonics seem totally sufficient. Higher amounts even suggest a drop in performance again. Interestingly, the sub-harmonic extensions do not show any improvements in the separation performance, although they cause a much more stable convergence during training. The dense harmonic extensions on the other hand show comparable good results in all settings.

	All		Drums		Bass		Other		Vocals	
Dataset	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR
Normal	**3.35**	**4.59**	4.34	5.03	**2.76**	**4.12**	**2.38**	**3.72**	4.73	5.40
Random	3.12	4.48	**4.89**	**5.36**	1.75	3.28	1.80	3.38	**4.76**	**5.44**

Table 4.5.: Median of SI-SDR and SDR for the proposed U-Net (5h+3d) trained with randomly generated mixtures of the MusDB18.

Another interesting result of the performed experiments is the effect of the randomly generated training mixtures on the separation performance. Table 4.5 compares the evaluation results of the proposed U-Net trained with the original MusDB18 against the results when trained with the randomly generated MusDB18 training set (see section 3.2). The validation and test set used in both trained models was the original MusDB18, without randomly generated mixtures. As described in section 3.2 the randomly generated dataset contains three times the amount of data points as the original dataset, to take advantage of the combinatorial possibilities of this data generation technique. Both experiments were trained under the standard settings with 3200 datapoints per epoch for a fair comparison. It can be seen the original MusDB18 dataset outperforms the randomly generated dataset in the overall performance (All: SI-SDR and SDR), which validates the assumption that the network takes advantage of the harmonic and rhythmic relationships between different sources. Astonishingly, the difference between the two evaluation results is much smaller than expected and the original dataset is even outperformed in the drum separation task with a relatively large margin as well as in the vocal separation task. On the other hand, the categories bass and other show poor results. This shows that mixing musical phrases randomly together could be a real alternative for the creation of new datasets.

During the design of the training procedure the results of different data augmentation sizes, as well as the combination of data augmentation with the randomly generated mixture dataset was evaluated. In general, all data augmentation techniques improve the separation performance. However, training only with normal data augmentation techniques (see section 3.2) does not exceed a benefit of more than $\approx 0.4\,\mathrm{dB}$ SI-SDR. On the other hand, the combination of normal data augmentation and randomly generated mixtures showed a boost of almost $1\,\mathrm{dB}$ SI-SDR.[1] This again proofs the benefit of the random mixture strategy.

[1]The data augmentation evaluation results can be seen in the appendix B.3

Furthermore, the effect of reducing the total amount of trainable parameters on the separation performance was evaluated by reducing the number of feature maps in each layer.[1] Both, the normal U-Net (U_{H+A}) and the densely-connected U-Net ($Dense_{H+A}$) were evaluated with 30, 9, and 3 million trainable parameters. The evaluation results are shown in the appendix in table B.4. The performance of the normal U-Net decreases by around 0.4 dB SI-SDR because of the feature map reduction. The densely-connected U-Net on the other hand shows a constant performance independent of the number of contained parameters. The overall SI-SDR only varies by 0.09 dB. The normal U-Net is outperformed in both cases with 9 and 3 million trainable parameters. This shows the predicted parameter efficiency of the densely-connected structure and correlates with the results of MMDENSELSTM, which yields the current State-of-the-Art results with only 1.3 million parameters [54]. Although it is parameter efficient, the densely-connected structure showed an enormous computational overhead during training due to its deep consecutive structure, which cannot take as much advantage of parallelization. It takes around six times as long to train the densely-connected architecture in comparison to the normal proposed U-Net under same the conditions.[1]

Finally, the effect of the PA-Net on the separated source signals was analyzed. Figure 4.4 shows the impact of the TF mask and the subsequent PA-Net post-processing on a single snare sound taken from a drum track of the MusDB18 test split. It can be seen, that the separated snare has a very slow attack, in comparison to the original sound. The snare is perceived like it contains a fade-in, similar to the hit with a brush. The PA-Net is able to restore the transient and re-align the 'noisy' mixture phase, to create the original impulsive attack. The sound quality of the separated drum signals is noticeably improved through the PA-Net post-processing, which is also visible in the evaluation results with a gain of 0.6 dB SI-SDR (see table 4.3). A similar effect is also noticed on the separated bass signals. The PA-Net is able to repair the 'diffuse' sound quality of the separated bass tracks and re-align the 'noisy' mixture phase. However, a quite large amount of aliasing is introduced by the PA-Net, which cancels out the positive effect. The bass evaluation results still rise with 0.25 SI-SDR. The modification of the amount of used similar/repetitive phrases K inside the self-attention mechanism, reveals that they provide the network access to additional useful information. However, only the separated drum tracks contain enough repetitive phrases to efficiently exploit this information. Here, the PA-Net trained with seven extra phrases ($K = 7$) showed an improvement of 0.9 dB SI-SDR in comparison to the PA-Net trained without any extra phrases ($K = 0$). The other source classes do not show such a high improvement, as they are not as repetitive as the drums. The results can be found in appendix B.5. In general, the PA-Net was able to reduce the amount of unwanted interfering source tracks, 'musical' noises, and other STFT artifacts quite well in all source classes. Unfortunately, this effect was replaced by aliasing artefacts in most source tracks, except the

[1] The amount of feature maps can be found in the appendix A.1

[1] Normal 30M: $\approx$ 16 hours, Dense 30M: $\approx$ 96 hours

drum tracks.

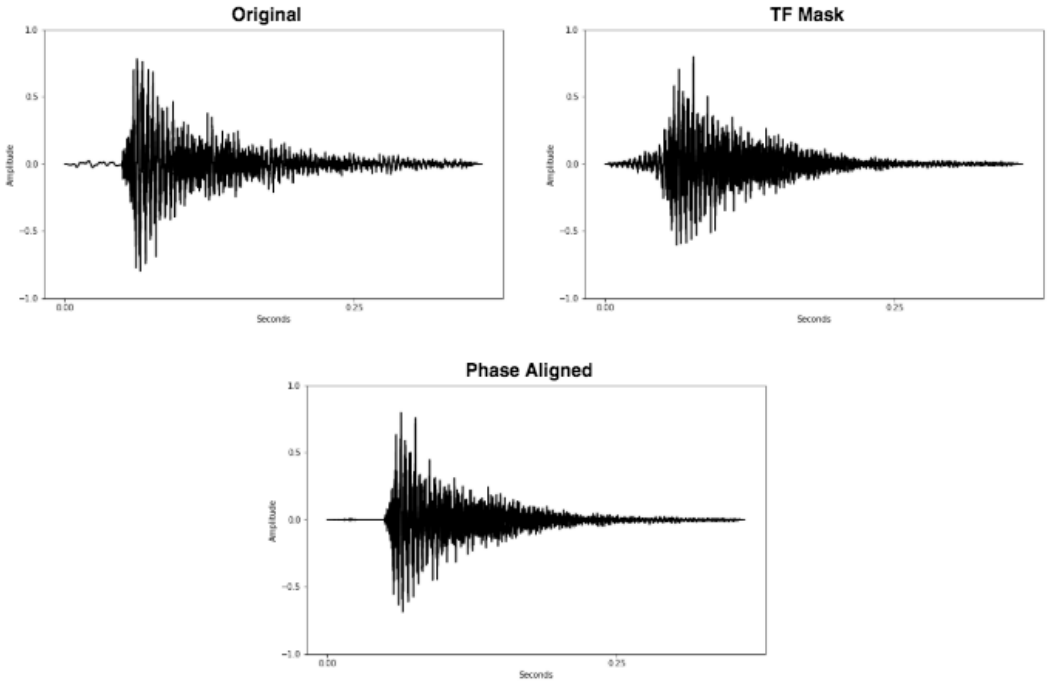

Figure 4.4.: A single snare sound taken from the MusDB18 test split. The upper left plot shows the original snare from the isolated source track. The upper right plot shows the same snare, separated with the proposed U-Net (MWF) from the mixture track. The lower plot shows the separated snare after being processed by the PA-Net.

A final word on the used evaluation metrics - During the evaluation it was noted that the gap between the evaluation results of two different models was almost always larger in the SI-SDR metric than in the SDR metric (see S vs. U_{H+} in table 4.2 as an example). Especially bad performing models are rated much higher in SDR than in SI-SDR. This confirms the findings of Le Roux et al. in [100] and justifies the predominant use of SI-SDR throughout this thesis. The adaptive FIR filter in the Museval toolbox is utilized to 'hide' bad performing separation models and distort the SDR results.

4.4. Discussion

Through the optimization of the internal flow of information, additional harmonic context information, and generative data augmentation the separation performance of Spleeter was improved with an average of 1.4 dB SDR. This demonstrates that the comparably simple U-Net architecture can still compete and even outperform State-of-the-Art models with more complex architectures, like MMDenseLSTM, or time-domain models like Demucs. The PA-Net post-processing has shown that the 'noisy' phase of a signal can be re-aligned afterward and was able to improve the performance by another 0.2 dB SDR. Furthermore, it was shown that repetitive phrases inside a song can be used to improve the overall sound quality, as long as the song contains enough repetitions. However, a convolutional network architecture, which prevents aliasing artefacts in the down/up-sampling path of the network still has to be found to take full advantage of this method.

All experiments have confirmed that the harmonic feature map extensions help the CNN to learn the exponentially scaled harmonic STFT relations, which yields better results and much more stable convergence. Interestingly, even the separation of the drums benefits from the harmonic information, although they do not contain much harmonic information. The ablation study showed that the normal harmonic extensions improved the separation results, while the sub-harmonic extensions stabilized the training convergence without improving the results. The dense-harmonic extensions showed the overall best results. One explanation for this observation could be that there are much more frequency bins in a spectrum that contain the energy of a harmonic than there are frequency bins that contain the energy of the fundamental frequency. In other words, there is a much higher density of harmonics in a complex spectrum than there are fundamentals. The sub-harmonic extensions are giving the network the ability to relate a harmonic back to its fundamental frequency. This helps to explain a much bigger amount of information of the spectrum than relating a fundamental frequency to its harmonic series like the normal harmonic extensions do. However, in the source separation task the latter is much more important: Identifying different instruments based on their harmonic series. This explains why the sub-harmonic feature map extensions help the network to find a much more stable solution space, which however does not improve the separation performance. The network can relate a harmonic in the spectrum to its fundamental frequency but it does not have access to the complete harmonic series of that fundamental, which would identify the source instrument. The dense-harmonic feature extensions solve this problem, which is probably the reason why they perform the best of all harmonic feature extensions.

The training on only randomly generated mixtures showed unexpectedly good results, indicating that harmonic and rhythmic relationships between sources are not so important to the separation performance as assumed. Analyzing the different instruments it can be seen that harmonic relationship between source instruments are utilized much stronger than the rhythmic relationships between them, because the sources, which in

general form the harmonic foundation (e.g. bass, piano, guitar,...) showed a bad performance when trained on randomly generated mixtures. Whereas the drum separation task even benefits from the bigger variety of randomly generated data. The vocal separation model trained with random mixtures showed slightly better results. Vocals have in general a higher verity and are not always part of the harmonic foundation (i.e. rap or spoken parts), which might explain why they benefit from the training on randomly generated mixtures. This is not a proof that randomly generated datasets are an alternative for real datasets. However, considering that there are $N^{J\tilde{T}}$ possible examples, this technique might be superior in some situations.[1] Although these mixtures are not equivalent to a completely new example, their amount is enormous. This way, easily accessible production music and loop libraries could be gathered and randomly mixed to generate much bigger datasets for source separation while avoiding copyright infringement issues. The randomly generated mixtures also showed excellent results when used as a data augmentation technique. The combination of normal data augmentation (i.e. re-pitching, time-stretching, etc.) and additional randomly generated mixtures shows an overall improvement of 0.56 dB SDR. Compared to Prétet et al. with 0.2 dB SDR [65] and Uhlich et al. with a maximum of 0.35 dB SDR [64], this shows the effectiveness of this combined augmentation.

Despite of the success of recurrent- as well as densely-connected-structures shown in many papers (see chapter 3.5), the U-Net architecture could not benefit from any of these extensions. Although the implementations were deduced from previous papers and also tested to some amount in initial experiments, they were still arbitrary to some extent and might only show their benefits in a large empirical study over the architectural implementation and hyper-parameterization of these structures, which exceeds the scope of this master thesis. Nevertheless the parameter efficiency of the densely-connected structure was shown.

Also, the multi-tasking U-Net structure proposed by Kandale et al. in [83] did not show their observed improvements compared to the dedicated U-Nets, when implemented with the here proposed U-Net (see table 4.2 M-U$_{H+}$). The multi-tasking architecture still showed acceptable results while only containing $\frac{1}{4}$ of trainable parameters and having the fastest training time of all models trained here.[2] Especially for bigger datasets with a much higher amount of sources J, the multi-tasking approach can be beneficial.

All trained models, even the ones with high amounts of trainable parameters, did not show 'classical' signs of over-fitting visible in the learning curves. Both the train- and validation-loss constantly decreased with each epoch. This might be due to the relatively high amount of dropout layers used. The more likely factor, however, is the high complexity of the source separation problem itself. The DNN has to learn

[1] Where N is the size of the original dataset in songs, J is the number of sources and $\tilde{T}$ is a symbolic number describing all possible time lags applied to the source tracks to create a new unique mixture.
[2] U-Net: 16 hours, M-U-Net: 4 hours

a sequence to sequence regression with both the input and output containing each $N \times F \times I = 512 \times 1024 \times 2 = 1,048,576$ floating-point values per data point. Considering that the final full augmented dataset contains around 22,000 data points, this might make 'classical' over-fitting of the training data impossible. Still, the test split of MusDB18, which contains only 50 songs and is used currently as the standard evaluation set in almost all papers, can lead to a severe model bias towards specific artists and genres. This was already partly shown in [63]. As a reminder, the complete MusDB18 dataset contains only around 10 hours of music, which is not at all close to any representative set of examples, covering the vast varieties of different genres existing in music. Since there are not many publicly available datasets, due to previously mentioned copyright issues, it is difficult to evaluate this model bias.

5. Conclusion

5.1. Summary

In this thesis, common deep learning-based music source separation methods were evaluated and examined on their shortcomings. It was found that the CNN/CRNN based TF masking approach, which is currently the most widespread method, makes two wrong assumptions. Firstly, CNNs, originally developed for image processing, are not able to efficiently utilize the harmonic relationships of music. Since the convolutional kernels expect a uniform data-structure (i.e. scaling does not change depending on the region), they are not well suited to learn the exponentially scaled harmonic relationships of a spectrogram. Secondly, the mixture phase is used to reconstruct the separated source signals, which causes a degraded sound quality and leads to a 'washed-out' sound character with corrupted transients.

Two unique solutions to these problems were developed. It was shown that the harmonic information can be made accessible to CNNs by simply restructuring the input data and aligning harmonically related frequency bins over the feature dimension. These harmonic extensions are very easy to implement, do not require any additional network parameters or cause a large computational overhead, and were shown to efficiently boost the separation performance. Furthermore, they lead to a faster and more stable training convergence, making it possible to train even deeper networks. The developed harmonic feature map extension layer can be attached to the input of any CNN/CRNN which operates on a spectrogram and will hopefully also be able to improve other MIR and audio-related deep learning tasks. Moreover, it was shown, that the 'noisy' mixture phase of the separated sources can be re-aligned by another waveform-domain network yielding a good approximation of the original phase. Moreover, a novel self-attention mechanism was proposed, which was able to average out unwanted interferences and artifacts by utilizing repetitive phrases in the separated source track. Although this approach did not improve all source classes equally, mainly due to aliasing problems, the sound quality of the drum tracks was noticeably better.

Further, the assumption, that the training dataset requires professionally produced music mixtures, with correct harmonic and rhythmic relationships between source instruments, was partly disproved. The generative mixing of source tracks showed very similar results to the original mixtures and should be at least used for data augmentation.

Although the achieved results surpass most State-of-the-Art results, one could argue that the used model is significantly larger in terms of contained parameters, which makes a direct comparison in some sense 'unfair'. Since audio source separation is a very active field of research, it will not take long until new DNN architectures achieve even better results. Rather, the point of this thesis was to show, that the design of the separation model can also benefit from incorporating music-specific context information. This was achieved by giving the network better access to the relevant information, which led to a better exploitation of the data. While many researchers base their improved performance on the adaptations of most recent deep learning techniques into their model, the here proposed ideas can be adapted to most music source separation DNN architectures to improve their performance. Even though both harmonic information and repetition were already exploited in previous model-based separation methods (e.g. NMF, KAM, etc.) their implementation inside the deep learning framework is unique.

As long as the available datasets remain small, the here proposed ideas help to improve the quality of the separated sources. In the longer run, raw waveform-domain networks, similar to WaveNet [101], will eventually outperform TF methods, as they scale better with additional data and computational power and contain no upper bound on their performance.

5.2. Future Work

Building on the results of this thesis, a unified and more compact separation model, which incorporates TF-masking, repetition based self-attention, and the waveform-domain post-processing into one single DNN is suggested. Since current deep learning frameworks already come with completely differentiable STFT and ISTFT implementations, such an architecture is possible in theory. This would combine the advantages of both methods. The sparse representation in the TF domain makes the separation easier and the final waveform representation can be used to synthesis a high quality source audio. The attention mechanism can further enhance the receptive field and provide the finale waveform domain network with additional information. Through a joint optimization, the network can learn to utilize the benefits of each representation. A symbolic representation is shown in figure 5.1.

Another aspect of research are perceptual studies over different separation approaches regarding the sound quality. Current State-of-the-Art networks like MMDenseLSTM and Demucs both achieve similar results in terms of SDR, although their introduced distortions are very different. On one side, the TF masking approach leads to a 'noisy' phase, 'musical' noises and other STFT artefacts. On the other side, the waveform-domain processing yields aliasing and a 'harsh' distortion. A perceptual evaluation of these distortions is necessary since current evaluation metrics like SDR and SAR do not provide enough information about the effect on the perceived sound quality.

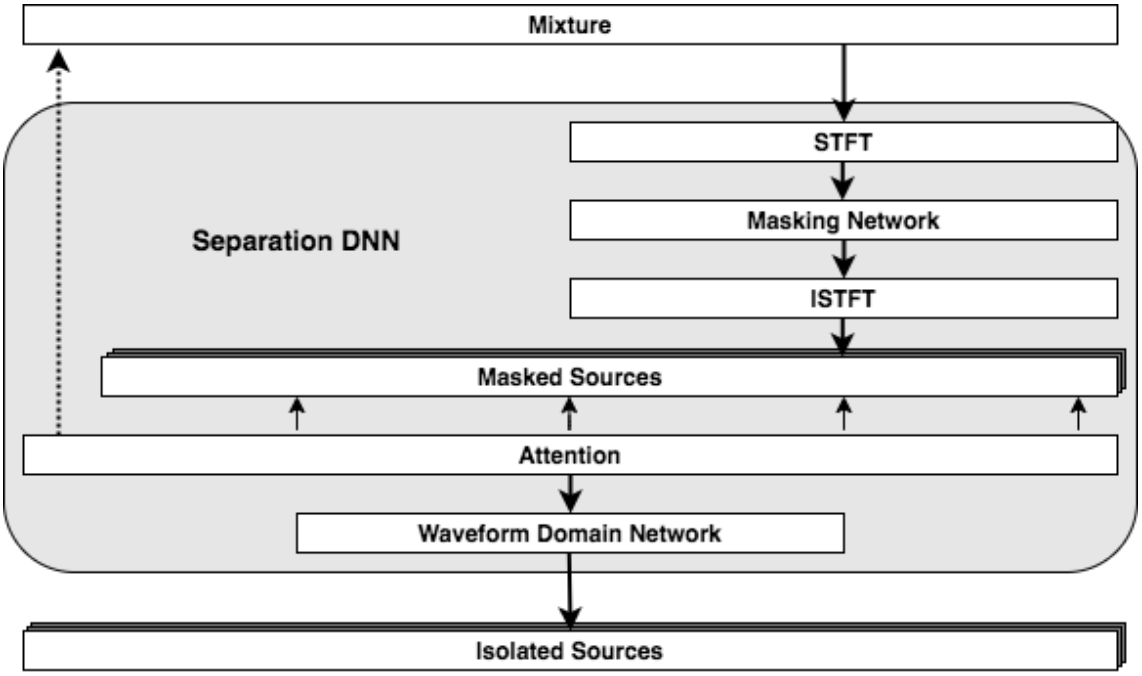

Figure 5.1.: Symbolic representation of a DNN using both the time frequency domain and the time domain for separation.

Furthermore, a perceptual study could also be used to develop new loss functions to train DNNs for source separation, which focus more on the human perceived sound quality. For example, Manocha et al. already proposed a perception based loss in [102], which consists of a DNN trained on the human perceived just-noticeable-difference between two sounds.

Finally and probably most importantly, much bigger public available datasets are required for music source separation, so that also researches, outside of big companies, like Spotify, Sony and Facebook, are able to generalize their proposed methods.

Bibliography

[1] Albert S Bregman. "Auditory Scene Analysis and the Role of Phenomenology in Experimental Psychology." In: *Canadian Psychology/Psychologie canadienne* 46.1 (2005), p. 32.

[2] Albert S Bregman. "Auditory scene analysis: Hearing in complex environments". In: (1993).

[3] Roger N Shepard. "Psychophysical complementarity". In: *Perceptual organization* (1981).

[4] Albert S. Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[5] Zafar Rafii et al. "An overview of lead and accompaniment separation in music". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.8 (2018), pp. 1307–1335.

[6] Nikolaos Mitianoudis and Michael E Davies. "Audio source separation of convolutive mixtures". In: *IEEE transactions on Speech and Audio processing* 11.5 (2003), pp. 489–497.

[7] Alexey Ozerov, Emmanuel Vincent, and Frédéric Bimbot. "A general modular framework for audio source separation". In: *International Conference on Latent Variable Analysis and Signal Separation*. Springer. 2010, pp. 33–40.

[8] A. Hyvärinne, J. Karhunen, and E. Oja. "Independent Component Analysis". In: *New Wiley Interscience* (2001).

[9] Shoji Makino. *Audio Source Separation*. Vol. 433. Springer, 2018.

[10] Guy J Brown and Martin Cooke. "Computational auditory scene analysis". In: *Computer speech and language* 8.4 (1994), pp. 297–336.

[11] Emmanuel J Candès et al. "Robust principal component analysis?" In: *Journal of the ACM (JACM)* 58.3 (2011), pp. 1–37.

[12] Po-Sen Huang et al. "Singing-voice separation from monaural recordings using robust principal component analysis". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2012, pp. 57–60.

[13] Zhuo Chen and Daniel PW Ellis. "Speech enhancement by sparse, low-rank, and dictionary spectrogram decomposition". In: *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE. 2013, pp. 1–4.

[14] Emmanuel Vincent, Rémi Gribonval, and Mark D Plumbley. "Oracle estimators for the benchmarking of source separation algorithms". In: *Signal Processing* 87.8 (2007), pp. 1933–1950.

[15] Aapo Hyvärinen and Erkki Oja. "Independent component analysis: algorithms and applications". In: *Neural networks* 13.4-5 (2000), pp. 411–430.

[16] Hans Fischer. *A history of the central limit theorem: From classical to modern probability theory*. Springer Science & Business Media, 2010.

[17] Aapo Hyvärinen. "New approximations of differential entropy for independent component analysis and projection pursuit". In: *Advances in neural information processing systems*. 1998, pp. 273–279.

[18] Ella Bingham and Aapo Hyvärinen. "A fast fixed-point algorithm for independent component analysis of complex valued signals". In: *International journal of neural systems* 10.01 (2000), pp. 1–8.

[19] Alexey Ozerov, Cédric Févotte, and Emmanuel Vincent. "An introduction to multichannel NMF for audio source separation". In: *Audio Source Separation*. Springer, 2018, pp. 73–94.

[20] Shoko Araki et al. "The fundamental limitation of frequency domain blind source separation for convolutive mixtures of speech". In: *IEEE Transactions on Speech and Audio Processing* 11.2 (2003), pp. 109–116.

[21] Alex Favaro, Aaron Lewis, and Garrett Schlesinger. "ICA for Musical Signal Separation". In: *N/A* ().

[22] Nicolas Gillis. "The why and how of nonnegative matrix factorization". In: *Regularization, optimization, kernels, and support vector machines* 12.257 (2014), pp. 257–291.

[23] Daniel D Lee and H Sebastian Seung. "Algorithms for non-negative matrix factorization". In: *Advances in neural information processing systems*. 2001, pp. 556–562.

[24] 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 (IEEE Cat. No. 03TH8684)*. IEEE. 2003, pp. 177–180.

[25] Cédric Févotte, Emmanuel Vincent, and Alexey Ozerov. "Single-channel audio source separation with NMF: divergences, constraints and algorithms". In: *Audio Source Separation*. Springer, 2018, pp. 1–24.

[26] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. "Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis". In: *Neural computation* 21.3 (2009), pp. 793–830.

[27] Tuomas Virtanen and Tom Barker. "Separation of Known Sources Using Non-negative Spectrogram Factorisation". In: *Audio Source Separation*. Springer, 2018, pp. 25–48.

[28] Alexey Ozerov et al. "Multichannel nonnegative tensor factorization with structured constraints for user-guided audio source separation". In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2011, pp. 257–260.

[29] Alexey Ozerov, Emmanuel Vincent, and Frédéric Bimbot. "A general flexible framework for the handling of prior information in audio source separation". In: *IEEE Transactions on audio, speech, and language processing* 20.4 (2011), pp. 1118–1133.

[30] Cédric Févotte and Alexey Ozerov. "Notes on nonnegative tensor factorization of the spectrogram for audio source separation: statistical insights and towards self-clustering of the spatial cues". In: *International Symposium on Computer Music Modeling and Retrieval*. Springer. 2010, pp. 102–115.

[31] Derry FitzGerald, Matt Cranitch, and Eugene Coyle. "Extended nonnegative tensor factorisation models for musical sound source separation". In: *Computational Intelligence and Neuroscience* 2008 (2008).

[32] Hirokazu Kameoka, Hiroshi Sawada, and Takuya Higuchi. "General formulation of multichannel extensions of NMF variants". In: *Audio Source Separation*. Springer, 2018, pp. 95–124.

[33] Neil Joseph Miller. *Removal of noise from a voice signal by synthesis*. Tech. rep. Utah University Salt Lake City Dept. of Computer Science, 1973.

[34] Guoning Hu and DeLiang Wang. "Monaural speech segregation based on pitch tracking and amplitude modulation". In: *IEEE Transactions on neural networks* 15.5 (2004), pp. 1135–1150.

[35] Adiel Ben Shalom et al. "Optimal filtering of an instrument sound in a mixed recording using harmonic model and score alignment". In: *Proceedings of the International Computer Music Conference (ICMC)*. 2004.

[36] Zafar Rafii, Antoine Liutkus, and Bryan Pardo. "REPET for background/foreground separation in audio". In: *Blind Source Separation*. Springer, 2014, pp. 395–411.

[37] Antoine Liutkus et al. "Kernel additive models for source separation". In: *IEEE Transactions on Signal Processing* 62.16 (2014), pp. 4298–4310.

[38] Paris Smaragdis, Gautham Mysore, and Nasser Mohammadiha. "Dynamic Nonnegative Models for Audio Source Separation". In: *Audio Source Separation*. Springer, 2018, pp. 49–71.

[39] Takuya Higuchi and Hirokazu Kameoka. "Unified approach for audio source separation with multichannel factorial HMM and DOA mixture model". In: *2015 23rd European Signal Processing Conference (EUSIPCO)*. IEEE. 2015, pp. 2043–2047.

[40] Francis R Bach and Michael I Jordan. "Learning spectral clustering, with application to speech separation". In: *Journal of Machine Learning Research* 7.Oct (2006), pp. 1963–2001.

[41] Antoine Liutkus, Roland Badeau, and Gäel Richard. "Gaussian processes for underdetermined source separation". In: *IEEE Transactions on Signal Processing* 59.7 (2011), pp. 3155–3167.

[42] Pablo A Alvarado, Mauricio A Alvarez, and Dan Stowell. "Sparse gaussian process audio source separation using spectrum priors in the time-domain". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 995–999.

[43] Hendrik Purwins et al. "Deep learning for audio signal processing". In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (2019), pp. 206–219.

[44] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[45] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[46] Naftali Tishby and Noga Zaslavsky. "Deep learning and the information bottleneck principle". In: *2015 IEEE Information Theory Workshop (ITW)*. IEEE. 2015, pp. 1–5.

[47] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[48] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[49] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[50] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[51] Jeffrey L Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[52] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[53] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[54] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation". In: *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE. 2018, pp. 106–110.

[55] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[56] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. "Multichannel audio source separation with deep neural networks". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.9 (2016), pp. 1652–1664.

[57] Alexandre Défossez et al. "Music source separation in the waveform domain". In: *arXiv preprint arXiv:1911.13254* (2019).

[58] Tak-Shing Chan et al. "Vocal activity informed singing voice separation with the iKala dataset". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 718–722.

[59] Chao-Ling Hsu and Jyh-Shing Roger Jang. "On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.2 (2009), pp. 310–319.

[60] Rachel M Bittner et al. "MedleyDB 2.0: New data and a system for sustainable data collection". In: *ISMIR Late Breaking and Demo Papers* (2016).

[61] Antoine Liutkus et al. "The 2016 signal separation evaluation campaign". In: *International conference on latent variable analysis and signal separation*. Springer. 2017, pp. 323–332.

[62] Zafar Rafii et al. "MUSDB18-a corpus for music separation". In: (2017).

[63] Ethan Manilow et al. "Cutting music source separation some slakh: a dataset to study the impact of training data quality and quantity". In: *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2019, pp. 45–49.

[64] Stefan Uhlich et al. "Improving music source separation based on deep neural networks through data augmentation and network blending". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 261–265.

[65] Laure Prétet et al. "Singing voice separation: A study on training data". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 506–510.

[66] Daniel Stoller, Sebastian Ewert, and Simon Dixon. "Adversarial semi-supervised audio source separation applied to singing voice extraction". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 2391–2395.

[67] Stefan Uhlich, Franck Giron, and Yuki Mitsufuji. "Deep neural network based instrument extraction from music". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 2135–2139.

[68] Po-Sen Huang et al. "Singing-Voice Separation from Monaural Recordings using Deep Recurrent Neural Networks." In: *ISMIR*. 2014, pp. 477–482.

[69] Andrew JR Simpson, Gerard Roma, and Mark D Plumbley. "Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network". In: *International Conference on Latent Variable Analysis and Signal Separation*. Springer. 2015, pp. 429–436.

[70] Pritish Chandna et al. "Monoaural audio source separation using deep convolutional neural networks". In: *International conference on latent variable analysis and signal separation*. Springer. 2017, pp. 258–266.

[71] Abhimanyu Sahai, Romann Weber, and Brian McWilliams. "Spectrogram feature losses for music source separation". In: *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE. 2019, pp. 1–5.

[72] Antoine Liutkus and Roland Badeau. "Generalized Wiener filtering with fractional power spectrograms". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 266–270.

[73] John R Hershey et al. "Deep clustering: Discriminative embeddings for segmentation and separation". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 31–35.

[74] Li Li and Hirokazu Kameoka. "Deep clustering with gated convolutional networks". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 16–20.

[75] Yi Luo et al. "Deep clustering and conventional networks for music separation: Stronger together". In: *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2017, pp. 61–65.

[76] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[77] Andreas Jansson et al. "Singing voice separation with deep u-net convolutional networks". In: (2017).

[78] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[79] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[80] Stylianos Ioannis Mimilakis et al. "Examining the Mapping Functions of Denoising Autoencoders in Singing Voice Separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2019), pp. 266–278.

[81] Romain Hennequin et al. "Spleeter: a fast and efficient music source separation tool with pre-trained models". In: *Journal of Open Source Software* 5.50 (2020), p. 2154.

[82] Jen-Yu Liu and Yi-Hsuan Yang. "Denoising auto-encoder with recurrent skip connections and residual regression for music source separation". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 773–778.

[83] Venkatesh S Kadandale et al. "Multi-task U-Net for Music Source Separation". In: *arXiv preprint arXiv:2003.10414* (2020).

[84] Fabian-Robert Stöter et al. "Open-unmix-a reference implementation for music source separation". In: (2019).

[85] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[86] Naoya Takahashi and Yuki Mitsufuji. "Multi-scale multi-band densenets for audio source separation". In: *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2017, pp. 21–25.

[87] Daniel Stoller, Sebastian Ewert, and Simon Dixon. "Wave-u-net: A multi-scale neural network for end-to-end audio source separation". In: *arXiv preprint arXiv:1806.03185* (2018).

[88] Yi Luo and Nima Mesgarani. "Tasnet: time-domain audio separation network for real-time, single-channel speech separation". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 696–700.

[89] Yi Luo and Nima Mesgarani. "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation". In: *IEEE/ACM transactions on audio, speech, and language processing* 27.8 (2019), pp. 1256–1266.

[90] Jaehoon Oh, Duyeon Kim, and Se-Young Yun. "Spectrogram-channels u-net: a source separation model viewing each channel as the spectrogram of each source". In: *arXiv preprint arXiv:1810.11520* (2018).

[91] Olga Slizovskaia, Gloria Haro, and Emilia Gómez. "Conditioned source separation for music instrument performances". In: *arXiv preprint arXiv:2004.03873* (2020).

[92] David Samuel, Aditya Ganeshan, and Jason Naradowsky. "Meta-learning Extractors for Music Source Separation". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 816–820.

[93] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[94] Rachel M Bittner et al. "Deep Salience Representations for F0 Estimation in Polyphonic Music." In: *ISMIR*. 2017, pp. 63–70.

[95] Jen-Yu Liu and Yi-Hsuan Yang. "Dilated convolution with dilated GRU for music source separation". In: *arXiv preprint arXiv:1906.01203* (2019).

[96] Alexandre Défossez et al. "Demucs: Deep Extractor for Music Sources with extra unlabeled data remixed". In: *arXiv preprint arXiv:1909.01174* (2019).

[97] 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.

[98] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. "The 2018 signal separation evaluation campaign". In: *International Conference on Latent Variable Analysis and Signal Separation*. Springer. 2018, pp. 293–305.

[99] Emmanuel Vincent et al. "First stereo audio source separation evaluation campaign: data, algorithms and results". In: *International Conference on Independent Component Analysis and Signal Separation*. Springer. 2007, pp. 552–559.

[100] Jonathan Le Roux et al. "SDR–half-baked or well done?" In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 626–630.

[101] Aaron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[102] Pranay Manocha et al. "A differentiable perceptual audio metric learned from just noticeable differences". In: *arXiv preprint arXiv:2001.04460* (2020).

Appendices

A. Hyperparameter Settings

Baseline Model and Proposed Model Architecture

This section provides a more detailed representation U-Net design as well as its hyperparameter settings. It ensures the reproducability of the achieved results.

The original Spleeter baseline model consists of six encoder and decoder stages with typical U-Net like skip and concatenate operations between each encoder/decoder pair. One encoder stage consists of a two-dimensional convolution with C filters of size 5×5 and a stride of 2×2. After the convolution, a batch normalization is performed over the channel axis followed by an Exponential Linear Unit (ELU) activation function. The decoder layers are designed in a very similar way. A transposed two-dimensional convolution with C filters of size 5×5 and a transposed stride of 2×2 is performed followed by an ELU activation function and a batch normalization over the channel dimension. To the lowest three decoder layer, a dropout with a chance of 50% is applied. After every decoder layer, the output is concatenated to the output of the convolution in the encoder side forming the typical U-Net structure. The lowest encoder layer doesn't utilize a batch normalization or an activation function. The last layer of the decoder outputs a single feature map which then gets processed by another two-dimensional convolution with two filters of size 4×4, no stride, and dilation of 2×2, which is followed by a sigmoid activation, creating a mask between $0 - 1$. The final output of this layer is then multiplied to the input of the model, resulting in an internal learned mask.

The proposed model doubles the number of layers from 13 to 26 by using two convolutional layers per stage instead of one. Each encoder stage consists of a down-sampling layer and a transition layer. The deepest transition layer is configured to have a dilation factor of 2 to increase the receptive field of the network. One down-sampling layer consists of a two-dimensional convolution, which increases the feature map size, followed by a batch normalization layer, a Rectified Linear Unit (ReLU), and a max-pooling operation with stride and pool size of 2×2. The transition layer consists of the same elements except for the max-pooling operation. The decoder of Spleeter is restructured and build up exactly symmetrical to the encoder, using a transposed convolution with a stride of 2×2 for up-sampling, followed by a batch normalization layer and a ReLU. The output of each encoder stage is concatenated to the output of the respective decoder stage. A dropout layer with a 50% dropout chance is used in the three lowest decoder blocks. The amount of feature map in the penultimate convolutional layer is re-scaled from one feature map to sixteen, as it might be disadvantageous to collapse all the information

into one feature map and then scale it up again in the last layer. The rest of the network remains the same as in the original model. An internal mask is created by using a sigmoid activation function in the last layer and multiplying it on the input tensor. The number of feature maps per stage for the original and proposed U-Net can be found in table A.1.

Model	Param.	Encoder	GRU Units	Decoder
Baseline	30	48 64 120 250 500 700	-	700 500 120 96 64 48
U-Net	30	48 64 96 160 300 600	-	600 300 160 96 64 48
U-Net GRU	30	48 64 96 160 300 600	700	600 300 160 96 64 48
U-Net Dense	30	68 51 171 128 488 378	-	378 189 189 79 79 32
U-Net	9	32 48 64 100 160 320	-	320 160 100 64 48 32
U-Net Dense	9	73 36 171 85 355 177	-	177 88 88 43 43 19
U-Net	3	20 30 50 70 90 150	-	150 90 70 50 30 20
U-Net Dense	3	64 27 117 49 229 97	-	97 48 48 24 24 12

Table A.1.: Overview of the number of feature map/number of filters used in the convolutional layers for encoder and decoder. For the densely-connected models, the number of feature maps after each block and after each down-sampling stage are shown.

	Dense Block 1			Dense Block 2			Dense Block 3			
Param.	N	M	θ	N	M	θ	N	M	θ	total
30M	5	8	0.75	15	8	0.75	30	12	0.775	65
9M	5	9	0.5	15	9	0.5	30	9	0.5	65
3M	4	9	0.425	10	9	0.425	20	9	0.425	49

Table A.2.: Hyperparameter settings for each densely-connected block. N: Number of layer per block, M: Number of filters per layer, θ: Downs-sampling factor at the end of each block.

Drum	Bass	Other	Vocal
1.49	2.51	1.0	1.49

Table A.3.: Sample weights for each source instrument in the M-U-Net.

B. Ablation Study Results

	All		Drums		Bass		Other		Vocals	
	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR
Std.	0.05	0.05	0.09	0.06	0.13	0.15	0.07	0.08	0.14	0.01

Table B.1.: The standard deviations of the SI- and SDR values calculated over five different training's with the proposed model with harmonic feature map extensions (5h+3d).

	All		Drums		Bass		Other		Vocals	
Harmonics	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR
U 0	2.96	4.29	4.04	4.65	2.16	3.61	2.17	3.31	3.91	5.13
U 3	3.31	4.54	4.36	4.84	2.49	4.00	2.34	3.54	4.72	5.51
U 5	3.23	4.51	4.60	5.17	2.75	4.14	2.37	3.52	4.51	5.41
U 9	3.23	4.41	4.29	4.74	2.62	3.95	2.26	3.45	4.05	5.37
U 5 + 1s	3.17	4.47	4.65	**5.25**	2.82	4.06	**2.41**	3.55	4.16	5.26
U 5 + 3s	3.21	4.56	4.28	4.96	2.20	3.81	2.33	3.69	4.65	5.43
U 9 + 5s	3.22	4.51	4.21	4.81	**2.97**	4.12	2.27	3.42	4.29	5.24
U 3 + 3d	**3.35**	4.56	**4.67**	5.13	2.71	3.91	2.30	3.63	4.68	4.68
U 5 + 3d	**3.35**	**4.59**	4.34	5.03	2.76	4.12	2.38	**3.72**	4.73	5.40
U 9 + 5d	3.34	**4.59**	4.27	4.86	2.65	**4.22**	2.22	3.58	**4.78**	**5.59**
S 0	2.45	3.94	3.58	4.56	2.13	3.51	1.66	3.17	2.53	4.39
S 5h+3d	**2.83**	**4.26**	**4.39**	**4.81**	**2.49**	**3.97**	**1.76**	**3.29**	**3.50**	**4.77**

Table B.2.: The median of Si-SDR and SDR for different harmonic extensions (s = sub-harmonics, d = dense-harmonics) with the proposed U-Net (U), as well as the on average best harmonic extension added to the original Spleeter architecture.

	All		Drums		Bass		Other		Vocals	
Augmentation	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR
0%	3.35	4.59	4.34	5.03	2.76	4.12	2.38	3.72	4.73	5.40
25%	3.71	4.71	4.54	4.89	3.22	4.35	2.59	3.86	4.76	5.76
50%	3.62	4.71	4.72	5.26	2.89	4.27	2.59	3.98	5.03	5.94
25% + 25%	**4.27**	**5.15**	**5.29**	**5.56**	**3.50**	**4.54**	**2.73**	**4.12**	**5.71**	**6.03**

Table B.3.: The median of SI-SDR and SDR for the proposed U-Net (5h+3d) trained with 25% and 50% interleaved augmented mixtures, as well as extra interleaved randomly generated mixtures (25%+25%). The number of total seen datapoints stays fixed at 3200 per epoch.

	All		Drums		Bass		Other		Vocals	
Model	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR
Normal 30M	**4.27**	**5.15**	5.29	**5.56**	3.50	4.54	2.73	4.12	**5.71**	6.03
Normal 9M	3.80	4.92	5.19	5.43	3.08	4.21	2.63	4.04	5.23	5.88
Normal 3M	3.88	4.85	5.18	5.29	3.39	4.41	2.32	3.83	5.12	5.80
Dense 30M	3.97	5.06	**5.43**	5.55	**3.70**	**4.67**	2.62	3.96	5.36	5.86
Dense 9M	4.06	4.98	5.28	5.47	3.14	4.53	**2.75**	**4.26**	5.68	**6.08**
Dense 3M	4.00	4.98	5.32	5.38	3.53	4.46	2.64	4.19	5.47	5.89

Table B.4.: Median of SI-SDR and SDR for the proposed U-Net (5h+3d) and the densely-connected U-Net trained on the augmented dataset. The results of GRU- and Dense-extensions are compared to the standard proposed U-Net. The GRU is trained on different time lengths.

	All		Drums		Bass		Other		Vocals	
Methode	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR	SI.	SDR
MWF	4.84	5.78	6.48	6.64	4.22	5.33	**3.49**	**4.78**	6.74	6.85
PA ($K = 0$)	4.94	5.76	6.20	6.50	4.36	**5.66**	3.43	4.68	6.76	6.82
PA ($K = 3$)	5.00	**5.86**	6.75	6.94	4.41	5.63	3.45	4.70	6.86	6.86
PA ($K = 7$)	**5.08**	5.83	**7.08**	**6.96**	**4.47**	5.63	3.40	**4.78**	**7.03**	**6.88**

Table B.5.: Median of SI-SDR and SDR for the PA-Net trained with different amounts of encoded similarity phrases K.