Graz University of Technology

Master Thesis

Efficient Single-Channel Music Source Separation with Deep Neural Networks

conducted at the
Signal Processing and Speech Communications Laboratory
Graz University of Technology, Austria

in co-operation with
sonible GmbH
Graz, Austria

by
Markus Huber, 01231594

Supervisors:
Univ.-Prof. Dipl.-Ing. Dr.mont Franz Pernkopf

Graz, April 20, 2020

This work was funded by the Austrian Research Promotion Agency (FFG) under grant 873144

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

date	(signature)

Acknowledgment

First of all, I would like to express my gratitude to the founders of sonible GmbH, Dipl.-Ing. Ralf Baumgartner, Dipl.-Ing. Peter Sciri and Dipl.-Ing. Alexander Wankhammer, for giving me the opportunity to work on this interesting and practically relevant topic, for letting me be a part of the amazing team that really distinguishes this company, for creating such a positive and motivational working environment and for their warm-hearted and inspirational nature.

Special thanks to Alexander Wankhammer, who always found time to give professional advice and to help me focusing on the essential when I lost track of what's important.

I am particularly thankful for the assistance provided by Dipl.-Ing. Christian Schörkhuber. This work greatly benefited from his expertise, and I sincerely appreciate the constructive suggestions and enthusiastic encouragement.

I gratefully thank Assoc.Prof. Dipl.-Ing. Dr.mont. Franz Pernkopf for his guidance, his great patience, the fast and valuable support throughout the entire development of this thesis, and his steady and straightforward help.

My thanks are also extended to the FWF Austria for supporting this work.

Finally, I wish to thank Barbara Weberndorfer for showing interest in my work and for her loving support at all times.

Abstract

The trend of utilizing deep learning techniques to tackle difficult signal processing problems has not spared the scope of single-channel source separation, and modern systems based on neural networks have indeed reached unprecedented levels of separation quality.

However, harnessing the power of these large-scale models in typical audio production environments, which frequently offer only limited computing resources while demanding quasi real-time processing, remains challenging. In order to utilize the power of deep neural networks on resource-constrained infrastructures, strategies and architectures have to be considered that ensure low computational requirements and memory footprint, while at the same time preserve (or even improve) accuracy.

This thesis sets out to examine viable solutions to both aspects of the problem within the context of musical audio mixtures, with a particular focus on singing-voice extraction. Various approaches to improve the performance of a state-of-the-art baseline system, the multi-scale multi-band DenseNet, are presented and discussed. These include architectural refinements regarding the multi-band structure, different training objectives, such as mask approximation, multi-task learning and deep clustering, as well as the exploitation and estimation of phase information, which allows for optimization in the time-domain. Specifically, instead of direct spectrogram estimation, experiments prove that using a deep clustering loss to approximate spectral masks results in a considerable performance increase over the baseline implementation.

Subsequently, the resource-efficiency of this system is addressed. It is shown that a significant reduction of the model size and its computational requirements can be achieved via an effective use of bottleneck layers and the inference of Mel-scaled masks. In addition, applying parameterized structured pruning of convolutional weights results in a further increase in efficiency.

Based on these findings, a high-quality source separation system can be obtained which is roughly 1.6 times smaller and 7.3 times more efficient than the state-of-the-art baseline while maintaining its separation performance. Moreover, on a 2018 Mac mini machine (i7 core), CPU inference times as low as 4 milliseconds were measured. Since this is well within the range of typical audio buffer block sizes, the real-time capability of this approach is thus confirmed.

Kurzfassung

Schwierige Fragestellungen der digitalen Signalverarbeitung werden immer häufiger mittels Deep Learning bewältigt. Nicht anders verhält es sich im Bereich der einkanaligen Quellentrennung (single-channel source separation), und in der Tat ist die erreichte Qualität von modernen Systemen, die auf neuronalen Netzwerken basieren, beispiellos. Es bleibt jedoch eine Herausforderung diese mächtigen und großen Modelle in üblichen Audioproduktionsumgebungen zum Einsatz zu bringen, da hier einerseits Echtzeitverarbeitung verlangt wird, andererseits aber nur begrenzte Rechenressourcen zur Verfügung stehen.

Um sich die Stärke neuraler Netzwerke auf ressourcenschwachen Infrastrukturen zu Nutze zu machen, müssen Strategien und Architekturen erörtert werden, die niedrige Rechenanforderungen sowie geringen Speicherbedarf aufweisen, gleichzeitig aber die Qualität des Systems nicht einschränken.

Diese Arbeit beschäftigt sich mit praktikablen Lösungen für beide Aspekte dieses Problems. Insbesondere liegt der Fokus auf der Extraktion der Gesangsstimme aus einem gegebenen Stück Musik. Aufbauend auf einem System auf dem neuesten Stand der Technik, dem multi-scale multi-band DenseNet, werden verschiedene Ansätze zur Steigerung dessen Performance präsentiert und diskutiert. Diese beinhalten Verbesserungen bezüglich der Architektur sowie hinsichtlich unterschiedlicher Trainingszielvorgaben. Ebenso wird die Nutzung und Schätzung von Phaseninformation behandelt, welche eine Optimierung im Zeitbereich ermöglichen. Experimente belegen dass der Einsatz von Deep Clustering in Kombination mit Masken- anstatt Spektrogrammschätzung zu beträchtlichen Leistungssteigerungen gegenüber dem Basismodell führt.

Anschließend wird die Ressourceneffizienz dieses Systems untersucht. Es wird gezeigt dass mit Hilfe von sogenannten Bottleneck Layern und Mel-skalierten Ein- und Ausgängen Größe und Rechenanforderungen des Modells signifikant verringert werden können. Außerdem kann die Effizienz durch strukturiertes Pruning der Gewichte weiter gesteigert werden.

Auf Grundlage dieser Ergebnisse kann ein qualitativ hochwertiges Quellentrennungssystem erzielt werden, welches um den Faktor 1.6 kleiner und um den Faktor 7.3 effizienter als das Ausgangssystem ist, ohne nennenswerten Einbußen hinsichtlich der Performance. Weiters werden auf einem Mac mini Rechner (i7 core, Late 2018) Inferenzzeiten auf der CPU von weniger als 4 Millisekunden gemessen. Dies ist durchaus im Bereich von üblichen Audiobuffer-Blockgrößen und bestätigt somit die Echtzeitfähigkeit dieses Ansatzes.

Contents

1 Introduction

1.1 Motivation

Given a mixture of source signals, how can these sources be isolated and recovered? This is the question that source separation systems try to answer. While the problem in general may arise in various domains, it is especially common with audio signals. Speech, music as well as environmental sounds frequently comprise a multitude of distinct acoustic components that are blended together, sometimes also extending between these three categories, yet in many situations only one of these components is of particular interest. Similarly, we sometimes might as well be interested in listening to the individual sources one by one. The concept behind music source separation is depicted in Fig. 1.1.

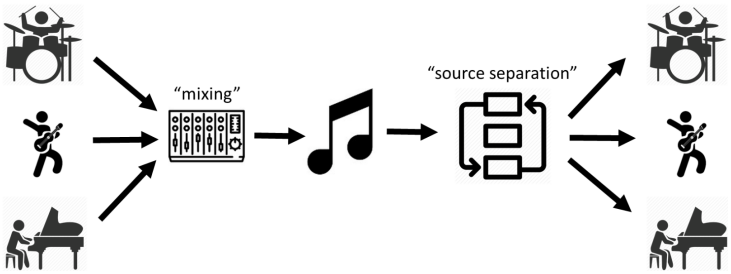


Figure 1.1: Illustration of the concept of music source separation [1].

Successful deployment of source separation methods plays a key role for a variety of tasks. Most importantly, many real-world applications depend on pre-processing via source separation, such as robust automatic speech recognition systems and music information retrieval tasks like chord or melody estimation. However, the usefulness of a stand-alone source separation system is evident: considering music signals alone, possible applications include creative content creation and remixing, automatic karaoke, audio post-production and remastering, upmixing e.g. stereo sounds to 5.1 surround as well as denoising.

Humans are notoriously good at solving this problem and the canonical example of a functional source separation system is what is called the cocktail party effect: the human brain's ability to distinguish a specific sound source against many others. The term stems from the exemplary scenario of chatting with one person at a cocktail party while not being distracted from other interfering conversations and/or overall high noise levels. A similar phenomenon can be identified in the case of musical signals, as it is possible to listen to a mix and focus only on a particular instrument. It is also probably due to this very circumstance that laypersons tend to underestimate the level of difficulty involved in finding ways to automate this task with computers. In fact, the separation of source signals poses an extremely hard problem to the digital signal processing community.

But what is it that makes this problem so challenging? In general, it is highly underdetermined, in the sense that usually the set of available mixture signals is smaller than the set of constituent source signals (with the latter therefore determining the overall system's number of degrees of freedom). For example, the majority of music mixtures is still limited to either one or two channels (mono or stereo), while most music typically features many more active contributing sources at a time. Underdetermined systems imply that the method used to unmix the signals has to be a non-linear one, and that it is typically hard to find a useful solution. Accordingly, hearing impairments in one ear significantly reduce the ability to solve the cocktail party problem. Additionally, acoustic signals may exhibit strong correlation, which makes it especially hard to distinguish between sources. For instance, the interplay between different musical instruments is usually based on a common tuning, key, tempo and rhythm, resulting in synchronous onset and offset times as well as strong overlaps in frequency. Moreover, professionally produced music mixtures are specifically designed to yield a coherent sound, and often are subject to heavy processing (e.g. via non-linear dynamic compression, equalization, reverberation, time-varying filters, etc.), complicating the source separation process even further. Finally, noise in the recordings may cause additional problems.

Nevertheless, over the course of several decades of research, much effort has gone into the development of source separation systems, and various approaches providing working solutions have been determined. More recently, with the advent of modern deep learning, new standards in performance have been set. Today, models based on deep neural networks (DNNs) attain extraordinary separation quality. Production-ready systems are already available and indeed the problem of source separation appears to be essentially solved.

This breakthrough in performance comes at a cost though, since modern DNNs involve an enormously large number of both parameters and computations. Hence environments that we would like to equip with high-quality source separation algorithms must be able to cope with the tough requirements such large models entail, and preferably include highly parallel processing capabilities like those provided by graphics processing units (GPUs). Clearly this is not feasible in many situations. For instance, hearing devices serve as a good example of such a platform with extremely limited computing resources. While it is generally possible to avoid on-device inference via cloud-computing based solutions, operating an internet connection may pose similar problems, not to mention the impracticality of real-time applications and the increased security risk. Typical audio production environments do not feature sufficient computational capacities to manage large-scale source separating DNNs either. For multi-track arrangements, which represent the vast majority of audio productions, available resources have to be shared across multiple audio tracks. Usually these tracks are also subject to individual processing such as dynamic range compression and equalization. Additionally, the audio callback is merciless and demands a buffer containing valid values at all times. If the audio buffer could not be filled in time, audible glitches may occur, possibly damaging the audio production or recording, the playback equipment or in the worst case a human ear. Since digital audio workstations are commonly run on consumer-grade desktop computers, employing overly expensive algorithms like DNNs on one or more audio tracks is basically infeasible, with respect to both memory and computing requirements. This is true even if a GPU is available, since it will usually be occupied with actual graphics rendering tasks or otherwise not feature enough RAM to fit both model and data.

However, taking advantage of the high separation quality offered by DNNs remains appealing nonetheless. The question arises whether these models are capable of resource-friendly real-time signal extraction within this context, while at the same time maintaining their separation performance.

1.2 Scope of the thesis

Over the last few years, the problem of utilizing the power of parameter-heavy DNNs within embedded systems has received much attention from the deep learning community. The fundamental question in this field of research revolves around the trade-off between performance and resource consumption in terms of computation, memory and energy. Much in the same spirit, this work is concerned with the practicality of deep learning for *efficient* source separation of music mixture signals within resource-constrained environments. The aim of this thesis is to find a good trade-off by separately exploring both aspects of the problem, performance and efficiency, in order to answer whether DNNs are capable of high-quality real-time music source separation on infrastructures with limited resources.

In particular, this thesis mainly focuses on recovering the *singing voice*. Vocals are complex in nature, exhibiting both harmonic and percussive signal components. For this reason it seems reasonable to conclude that solutions which work well for singing voice extraction will also generalize nicely to other musical source types.

Moreover, only the *single-channel* case is considered. In contrast to mixture signals which typically are available in multi-channel formats, musical instruments are frequently recorded using only one microphone. Though targeted on a single instrument, these recordings often capture additional sounds such as background noises or spill from accompanying instruments, and thus call for post-processing via source separation methods. On the other hand, microphone positions of multi-channel music signals are usually unknown and will also vary between different recordings, in addition to exhibiting sometimes severe phase issues. This prevents any pre-processing via beamforming techniques and also makes it hard for data-driven models to properly exploit multi-channel information. Another reason for limiting the scope of this work to the single-channel case is the very straightforward extension of DNNs to handle multi-channel inputs and outputs.

Additionally, many situations require *real-time* processing of audio signals. As outlined in the previous section, processing of audio during playback is time-critical and, sticking to the example of audio production environments, is necessary for on-the-fly editing. This requirement is especially hard to reconcile with the increased computational demands of large-scale neural networks.

1.3 Organization of the thesis

Following this introduction, Chapter 2 discusses characteristic differences between speech and music signals, and gives a comprehensive overview of various approaches to tackle the problem, including a section on how to evaluate the performance of source separation algorithms as well as a section on frequently used pre- and post-processing methods. In addition, a summary of related work is provided, with regard to both more traditional model-based as well as recent deep learning based approaches towards music source separation, including considerations concerning the architectural design, training objectives and also efficiency.

Rather than starting out from scratch, this work takes a rather experimental approach and uses a state-of-the-art baseline model, the so-called multi-scale multi-band DenseNet (MMDenseNet), as a sound starting point. After discussing details of the MMDenseNet topology and the experimental setup in Chapter 3, Chapter 4 considers the efficacy of several possible extensions in order to improve upon this baseline with regard to performance. These include both architectural alterations as well as modifications relating to the optimization. Since they are not necessarily disjoint, the approaches are covered in a successive manner and results are presented intermittently.

Continuing from the insights gained in Chapter 4, measures towards reducing the computational requirements and the memory footprint of the model are subsequently investigated in Chapter 5. Again, the particular modification's concept and the corresponding results are presented alternatingly.

Finally, Chapter 6 concludes the thesis with a summary and a discussion of the findings. Also, interesting open questions to be answered in the course of future work are addressed.

2 Overview and Related Work

After the general introduction to the problem of source separation and the statement of motives and intentions of the thesis in Chapter 1, this chapter provides the reader with a comprehensive overview of how one can go about solving the task. A thorough understanding of the mixture and target signals' nature is crucial for the design of a source separation algorithm. The characteristic structure of musical sources as well as the interplay between them *have* to be utilized in order to enable successful separation, and are described in Section 2.1.

Additional prerequisites for the development of a source separation system include a formal description of the problem and a way to evaluate an algorithm's separation performance, which are given in Sections 2.2 and 2.3, respectively.

Although this work focuses on data-driven methods derived from deep learning and their usefulness when computing resources are limited, it is nevertheless sensible to review other, more traditional approaches since these can provide valuable insights and outline desirable properties for a source separation algorithm. A comprehensive overview of some model-driven systems is provided in Sections 2.5.1 up to 2.5.3.

Subsequently, Section 2.5.4 discusses some of the recent advances in (music) source separation based on DNNs with regard to used architectures as well as training procedures, and equips us with the necessary information to choose a suitable baseline model as a starting point for this work.

2.1 Characteristics of music signals

Music signals contain a lot of distinct structure that distinguishes them from speech and environmental sounds, but also serves to differentiate between various instruments or instrument groups. Clearly, the separation of a target source depends on successfully exploiting these characteristic patterns in a reasonable manner. To this effect, the target signal must of course be properly identified. It is important to point out that a musical source may be defined in different ways, including single instruments, groups and families of instruments (e.g. a choir, or string and wind instruments) and more abstract categories, for example functions in a particular piece of music, such as the rhythm section or a division between lead and accompaniment. The characteristics of four different musical sources - singing-voice, electric bass, drum kit and electric guitar - are exemplified in Fig. 2.1 via their spectrograms. Another important issue in music source separation that is frequently overlooked is the bias towards western (pop) music. Other genres of music may benefit from different descriptors and criteria, and structural aspects may be more or less pronounced depending on the type of musical material. In general though, the following characteristics can be attributed to musical signals and may be exploited by source separation algorithms.

Harmonicity In the context of source separation, harmonicity corresponds to the notion of a sonic event's acoustic periodicity. Considering the magnitude spectrogram of a musical signal, harmonicity describes the strength and the relation between its "horizontal" components.

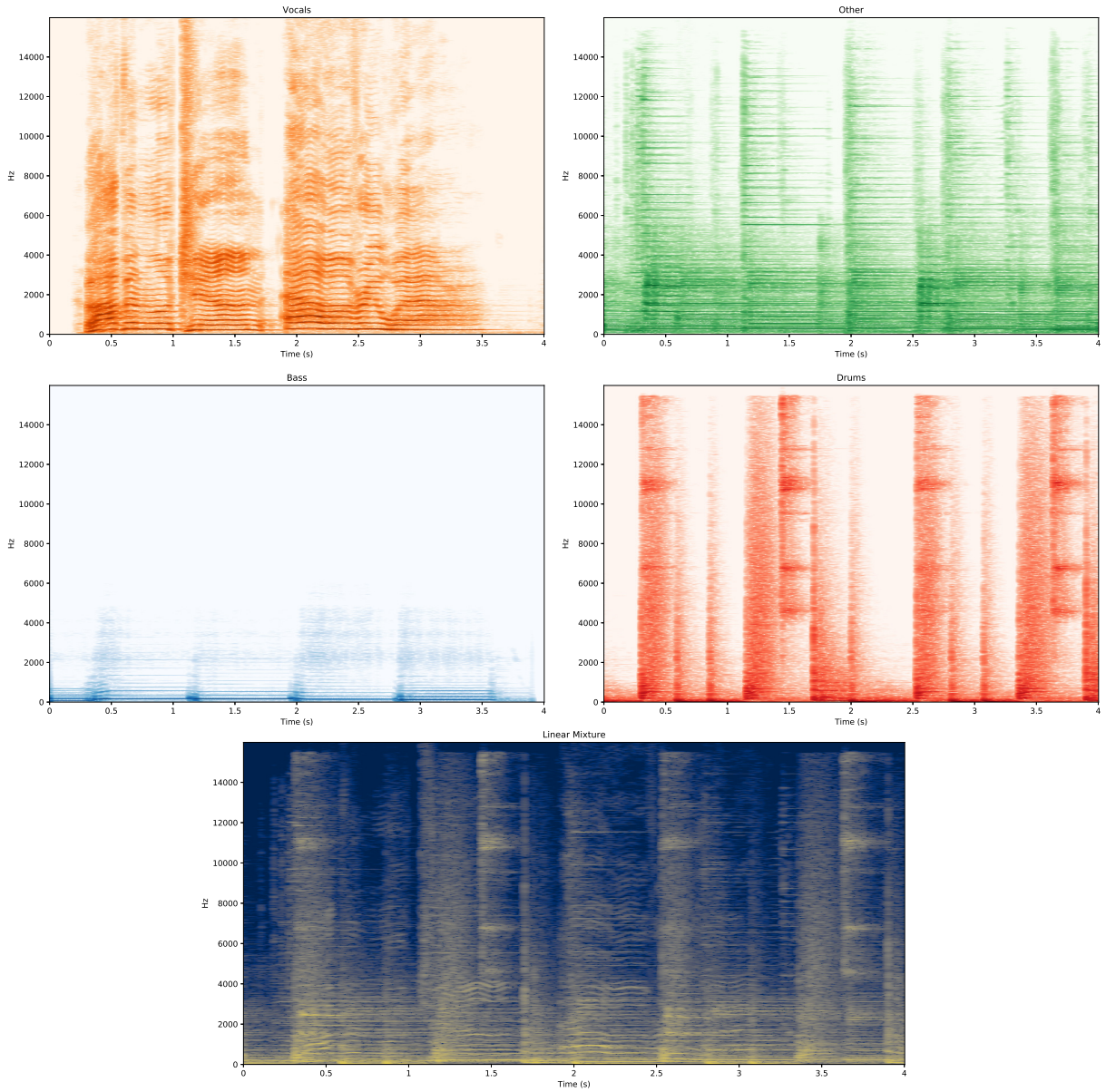


Figure 2.1: Magnitude spectrograms of the 4 different musical sources constituting the first 4 seconds of the song 'Music Delta - Beatles', taken from the musdb18 corpus, and their linear mixture. The source titled "Other" corresponds to an electric guitar in this case.

These components are highly structured and follow similar trajectories - a property which is often called the *common fate*. The lowest frequency, usually termed the *fundamental frequency* or f_0, determines the *pitch* of a musical source. The pitch itself can already be quite useful for identifying certain instruments. For instance, the size of an instrument is normally inversely proportional to the pitch it produces. Similarly, variations in pitch convey a lot of information, e.g. bowed string instruments typically exhibit a characteristic "jitter" in terms of pitch. But more information can be deduced from the related horizontal components with higher frequencies, which are called *overtones*, *partials* or *harmonics*. The latter term usually implies that these components are close to integer multiples of the fundamental frequency, which results in very clear and unambiguous tones as well as pitch perception (hence the name harmonics). Note that the pitch thus may also be derived from the distance between the harmonics: the brain may perceive a certain pitch even if the corresponding fundamental frequency is not at all present in the audio signal. Western music also frequently plays "in harmony", meaning that

the ratios of f_0s of notes played at the same time are close to small integer ratios. For instance, a C-major chord consists of the concurrent notes C, E and G, whose f_0 have a relationship of $C : E : G = 4 : 5 : 6$. This leads to a large overlap in frequency across each of the notes, but often also across multiple sources, which makes the separation process even more difficult. Inharmonic, and thus non-integer, relationships between partials produce less clean sounds, however the results may still be predominantly tonal. Typical examples of inharmonicity are sounds produced by oscillating bells, tubes, rods and also strings (for example bass strings of pianos). The location, relation and relative strength of partials, which may be summarized by the notion of a *spectral envelope*, strongly determines what is known as the "timbre" of a sound (although the word timbre has no useful definition, except for describing a "perceptual quality in addition to loudness, pitch and duration" [2]). As such it is a very salient feature for the recognition of a particular instrument.

Percussivity In addition to horizontal components, spectrograms of musical sources may exhibit vertical components, indicating that a wide range of frequencies is active at the same time. These are associated with percussive sounds, which typically show highly transient behavior and can be localized in time fairly accurately. While containing little or no pitch and harmonic content, percussive sounds may nonetheless feature distinct patterns in their spectral envelopes, for instance due to prominent resonances, that carry valuable information about an instrument's nature. The spectrogram of the drum track presented in Fig. 2.1 illustrates the character of 3 different percussive sounds very clearly, with a bass drum sound after 2 seconds, a snare drum after approximately 1.125 seconds, and a tambourine hit shortly before second 1.5.

Most signals however are actually a combination of both harmonic and percussive elements. Sounds from a singing voice serve as primary examples of this category. The vocal folds determine the pitch and carries voiced, predominantly harmonic, speech sounds. On the other hand, many consonants like the plosives /p/ and /t/ or fricatives such as /s/ and /f/ have a percussive nature. In addition, all sounds are given a distinct frequency pattern via the adaptive filtering caused by resonances in the vocal cavity. An example can be seen in the vocal spectrogram of Fig. 2.1, which exhibits a glottal stop followed by the voiceless and strikingly percussive consonant /k/ shortly after 1 second of the song have passed. Furthermore, this behavior is especially prominent during the attack of many musical sources. Most instruments have a kind of noisy onset. For instance, these may be caused by the hammer of a piano beating the strings, the initial airy sounds stemming from wind instruments, or the plucking, picking or strumming of strings. Indeed, the attack transient is extremely important for humans in order to identify various isolated musical sources [3]. Initial percussive sounds as well as the absolute and relative rise times of partials and their evolution over time are crucially informative when distinguishing instrumental sounds.

Repetition Music needs repetition to be comprehensible. Many forms of repetition in music can be discovered, for example repeating long-term structures of arrangements that divide musical parts into e.g. chorus and verse or exposition and recapitulation. But also shorter harmonic, melodic and rhythmic structures are essential to an enjoyable and understandable musical piece, such as motifs, hooks, themes, phrases, riffs and grooves. Many genres of music, e.g. electronic dance music, draw heavily on repetitive elements.

Sparsity Another characteristic of music signals is that they are particularly sparse in the sense that the majority of time-frequency (TF) bins carry very little energy. Though at first this does not seem overly beneficial in detecting particular sources, separation models may take advantage of this property by introducing e.g. sparse priors in order to overcome indeterminacies of the problem description (see Section 2.2).

2.1.1 Difference to speech signals

Although music and speech share a range of common audio processing tasks, such as source separation and automatic transcription, the two domains of sound are fundamentally different in several aspects. One key difference results from the restrictions that naturally come into effect in the case of speech sound production. The vocal apparatus operates a fixed set of speech producing organs, which is also subject to physiological limits: the vocal tract length, for example, stays within certain bounds across all human beings, albeit slightly varying dimensions for different people. Similarly, the set of formants (combinations of resonance frequencies within the vocal tract) is limited and follows a certain distribution - common languages would not even be conceivable if this was not the case. Of course, this is also true independently of any specific language. Music on the other hand has an extremely wide range of origins and thus may contain manifold and inconsistent content. Different types of instruments exhibit distinct structure in the time-frequency domain. The variable importance of frequency areas e.g. can be seen in Fig. 2.1, with the bass containing no energy at all above 5 kHz, the snare drum hit occupying the whole spectral range, and the emphasis of the singing-voice on the frequency bands changing with each phone and note. Moreover, the variance of a sound's character within a certain class of instruments may be significantly high too, and although the concept of e.g. a piano instrument imposes constraints on its design, these are much more relaxed compared to the biological and physiological limitations for the vocal apparatus.

In fact, the origin of a musical sound may be much more diverse and might even be hard to determine. While the properties described so far are inherent to music signals, musical sources may experience heavy transformation or manipulation, completely changing the original sound's character and structure. An electric guitar signal running through a multitude of effects (like e.g. a fuzz, wah, chorus, reverse delay, vibrato, and a pitch-shifting reverb, to name a few), captured through a particular microphone which has been precisely positioned in front of the speaker of a certain amplifier and mixed with a modified version of its direct-input (DI) signal, may serve as an excellent example. Other (maybe less drastic) examples showcasing the wide variety of musical sources include the recording of an instrument in a room with distinct acoustics or an entirely software based instrument. Additionally, the mixing process frequently imposes heavy modifications onto the sources, usually in order to account for this diversity and to obtain a coherent mixture signal. All of this is not very typical for speech sounds.

Another difference between speech and music signals is their different objective. While speech usually aims for intelligibility, the general objective of music is to evoke a certain aesthetic quality. This is also reflected in the frequency ranges both types of sounds are usually associated with. Accordingly, traditional telephone transmission technology offers only about 4 kHz of bandwidth (roughly 300 Hz - 3.4 kHz) which suffices for most voice applications, whereas music typically occupies the full audible range from 20 Hz to 20 kHz - even if it is an a-capella spoken-word performance. The human voice plays an integral part in many music cultures, however, the singing-voice fundamentally differs from spoken language. Although both may exhibit e.g. integer harmonic relationships, the usual goal for the singing-voice is to have a pronounced pitch, and very clear and clean harmonic structure, often in a polyphonic context and in consonance with other instruments. By contrast, the structure of speech is governed by a plethora of linguistic features, ranging from articulatory and grammatical to semantic and pragmatical considerations.

Finally, differences in loudness and intensities can be determined. For example, the vocal tract is highly damped, resulting in a rather low crest factor around the order of 12 dB [4]. Depending on the source type, the crest factor of music signals can be quite a lot higher and more unsteady.

2.2 Formal description of the problem

Blind source separation can be described using the notion of a mixture system. The task is then to estimate the N unknown source signals $s_n(t)$ given M observed mixture signals denoted by $x_m(t)$ which have been mixed via some mixture system $\boldsymbol{A}$. In the most simple form of linear instantaneous mixtures with time-invariant gains a_{mn}, $x_m(t) = \sum_{n=1}^{N} a_{mn} s_n(t)$, the problem can be conveniently expressed as a linear system

$$\boldsymbol{x}_t = \boldsymbol{A} \cdot \boldsymbol{s}_t + \boldsymbol{n}_t \tag{2.1}$$

with $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ denoting the mixing system, $\boldsymbol{x}_t = [x_1(t) \cdots x_M(t)]^T$ and $\boldsymbol{s}_t = [s_1(t) \cdots x_N(t)]^T$ containing the mixture and source signals at time t, respectively, and $\boldsymbol{n}_t = [n_1(t) \cdots n_M(t)]^T$ corresponding to an *optional* noise term introduced by the sensors capturing the mixtures.

However, a mixture system may not only describe time-invariant gains and can be more intricate. Input signals may be composed of attenuated and time-delayed sources

$$x_m(t) = \sum_{n=1}^{N} a_{mn} s_n(t - t_{mn}) \tag{2.2}$$

or of sources convolved with a certain time-invariant transfer function or filter:

$$x_m(t) = \sum_{n=1}^{N} \sum_{k=-\infty}^{\infty} a_{mnk} s_n(t - t_{mnk}) = \sum_{n=1}^{N} a_{mn}(t) * s_n(t) \tag{2.3}$$

This convolution filter could even be time-variant. The exact formulation depends on the application, which - along with the formulation of the evaluation, see Section 2.3.1 - defines the different forms of distortion that are allowed between the estimated and the true source. For instance, extracting clean signals from moving sources in a reverberant room would require a description using a time-variant source-to-microphone convolutive filter, though it might be extremely hard or even impossible to identify the original sources. Short observation durations in combination with long mixing filters, for example, raise serious technical difficulties. In the noiseless determined case, $\boldsymbol{x}_t = \boldsymbol{A} \cdot \boldsymbol{s}_t$ with $M = N$, the task can be framed as the search for the linear time-invariant demixing system $\boldsymbol{W} = \boldsymbol{A}^{-1}$ which is able to recover the sources simply using $\boldsymbol{s}_t = \boldsymbol{W} \cdot \boldsymbol{x}_t$. However, this is not possible in the underdetermined case ($M < N$), which has infinitely many solutions forming an affine space.

Irrespective of the description though, sources can in general only be recovered up to a permutation and arbitrary gains. The problem always suffers from inherent indeterminacies, which cannot be overcome without introducing prior knowledge or assumptions about the intrinsic properties of the source signals and/or the mixing system. In this sense, source separation is never truly "blind".

Although the spatial (e.g. stereo) image carries a lot of useful information with regard to the separation of the sources, many practical scenarios and applications require a restriction to single-channel mixtures ($M = 1$). This work is concerned with the single-channel case which is always underdetermined (since at least 2 sources are necessary for the problem to be relevant). In particular, the vocal and accompaniment sources act as the desired target signals. Furthermore, the focus of this thesis is to tackle this problem using a data-driven supervised deep learning method, and the considered mixture model is thus limited to the instantaneous linear model of the used dataset (see Section 3.3.1).

2.3 Performance evaluation

Finding adequate criteria for describing the quality of a source separation algorithm is an active field of research, and perceptual impressions often do not correlate well with objective evaluation procedures that strive to represent performance using one or more numeric metrics [5], [6]. It is clear though that a simple calculation of quality metrics, which ideally incorporate perceptual information, is highly desirable and eases the process of algorithm development by alleviating the need to perform cumbersome subjective listening tests. Additionally, it is not obvious how to design such perceptual evaluations [7]. Other interesting different directions of research are concerned with the issue of how to evaluate separation quality without references [8], and how to successfully crowdsource the evaluation process [9].

One perceptually motivated framework to objectively evaluate performance is *PEASS* [10]. However, the so-called *BSS Eval* metrics [11], despite facing some criticism from the research community (e.g. [12]), still remain the most widely used criteria in the source separation literature, which, for the sake of comparability to other work, have also been adopted in this thesis. These offer a rather interpretable and well established quantitative set of energy-based measures - though this very fact makes it hard to overcome or improve them. However, it is important to keep in mind that these measures do not guarantee a totally fair comparison between different source separation systems and neither do impeccably reflect their performance power.

2.3.1 BSS Eval metrics

The underlying reasoning behind the BSS Eval metrics is that the estimated source signal $\hat{\boldsymbol{s}}_n = [\hat{s}_n(t_1) \cdots \hat{s}_n(t_T)]^T$, with t_T denoting the last time step of the series, can be decomposed into several distinct terms, either corresponding to sounds associated with the true source $\boldsymbol{s}_n = [s_n(t_1) \cdots s_n(t_T)]^T$, sounds from other interfering sources $\boldsymbol{e}_n^{interf}$, components due to filtering or spatial errors $\boldsymbol{e}_n^{filt}$, or artifacts $\boldsymbol{e}_n^{artif}$ such as "gurgling" noise possibly introduced by the algorithm:

$$\hat{\boldsymbol{s}}_n = \boldsymbol{s}_n + \boldsymbol{e}_n^{filt} + \boldsymbol{e}_n^{interf} + \boldsymbol{e}_n^{artif} \tag{2.4}$$

For simplicity and notational convenience, channel indices have been discarded since only the single-channel case is of interest to this thesis. The decomposition into these distinct signal components is based on orthogonal projections of the estimate $\hat{\boldsymbol{s}}_n$ onto subspaces spanned by the respective signals corresponding to these components. Additionally, the signals used to form these subspaces may allow some kind of distortion introduced by the estimate. In the simplest case of time-invariant gains as a form of allowed distortion, the error due to the presence of other sources, $\boldsymbol{e}_n^{interf}$, can be obtained by projecting the estimate $\hat{\boldsymbol{s}}_n$ onto the subspace spanned by all source signals $\boldsymbol{s}_{n\prime \neq n}$ other than the target source $\boldsymbol{s}_n$. This is done by finding the corresponding projector $P_{n\prime}$, usually via a least squares solution. However, for the calculation of this projector the intercorrelations between the sources have to be accounted for as well. After the decomposition has been found, energy-based ratios can be formed which represent the actual evaluation criteria.

The concept of the BSS Eval metrics is thus to compute the actual metrics after matching the estimates to the references considering some formed of allowed distortion. In the music source separation community, convolutive mixtures are the most common case, and allowing linear distortion filters provides some robustness against linear mismatches. As pointed out in [12] and [13] though, this kind of matching can significantly overestimate an algorithm's performance if the distortion filters are assumed to be time-varying. Therefore, only time-invariant filters have been considered in this thesis.

Implementations of the BSS Eval criteria typically assume such distortion filters to have a finite impulse response (FIR) of tap-length $L = 512$. Thus, in order to find the error term $\boldsymbol{e}_n^{filt}$, the estimated source is projected onto the subspace spanned by delayed versions of the target reference signal $\boldsymbol{s}_n^\tau = [s_n(t_1 - \tau) \cdots s_n(t_T - \tau)]^T$, with $\tau \in [0, L-1]$ and $\boldsymbol{P}_n^L$ denoting the corresponding least-squares projector:

$$\boldsymbol{e}_n^{filt} = \boldsymbol{P}_n^L \, \hat{\boldsymbol{s}}_n - \boldsymbol{s}_n \tag{2.5}$$

Similarly, the error term associated with the interference of other sources may be decomposed using the projector $\boldsymbol{P}_N^L$, which projects onto the subspace spanned by delayed versions of all references $\boldsymbol{s}_n^\tau$, with $n \in [1, N]$ and $\tau \in [0, L-1]$.

$$\boldsymbol{e}_n^{interf} = \boldsymbol{P}_N^L \, \hat{\boldsymbol{s}}_n - \boldsymbol{P}_n^L \, \hat{\boldsymbol{s}}_n \tag{2.6}$$

The amount of artificial noise $\boldsymbol{e}_n^{artif}$ may then be thought of as a residual signal.

$$\begin{aligned} \boldsymbol{e}_n^{artif} &= \hat{\boldsymbol{s}}_n - \boldsymbol{s}_n - \boldsymbol{e}_n^{filt} - \boldsymbol{e}_n^{interf} \\ &= \hat{\boldsymbol{s}}_n - \boldsymbol{P}_N^L \, \hat{\boldsymbol{s}}_n \end{aligned} \tag{2.7}$$

Now separate performance criteria in the style of the signal-to-noise ratio can be defined using energy ratios of the distinct components: the source-to-interference ratio (SIR), the source-to-artifacts ratio (SAR) and the image-to-spatial-distortion Ratio (ISR). Note that the ISR incorporates filter distortions of the target source and although the expression "image-to-spatial-distortion" is rather meaningless in the single-channel case, it is kept for consistency with the usual terminology. The relationships of the BSS Eval signal components is represented graphically in Fig. 2.2. Additionally, the overall error in the estimate is quantified in the source-to-distortion Ratio (SDR), which includes all error components. These ratios are expressed in decibels (dB):

$$SDR_n = 10 \log \left(\frac{\|\boldsymbol{s}_n\|^2}{\|\boldsymbol{e}_n^{filt} + \boldsymbol{e}_n^{interf} + \boldsymbol{e}_n^{artif}\|^2} \right) \tag{2.8}$$

$$ISR_n = 10 \log \left(\frac{\|\boldsymbol{s}_n\|^2}{\|\boldsymbol{e}_n^{filt}\|^2} \right) \tag{2.9}$$

$$SIR_n = 10 \log \left(\frac{\|\boldsymbol{s}_n + \boldsymbol{e}_n^{filt}\|^2}{\|\boldsymbol{e}_n^{interf}\|^2} \right) \tag{2.10}$$

$$SAR_n = 10 \log \left(\frac{\|\boldsymbol{s}_n + \boldsymbol{e}_n^{filt} + \boldsymbol{e}_n^{interf}\|^2}{\|\boldsymbol{e}_n^{artif}\|^2} \right) \tag{2.11}$$

Evaluations are usually performed for a whole audio track, which also ensures that the distortion filter stays time-invariant. Nonetheless, the power of the sources typically varies over time. This can be taken into account by windowing the signal components $\boldsymbol{s}_n$, $\boldsymbol{e}_n^{filt}$, $\boldsymbol{e}_n^{interf}$ and $\boldsymbol{e}_n^{artif}$ before building the final *local* performance measures. Finally, the median value over all window frames can be computed in order to yield a single numeric value for each of the metrics for each track.

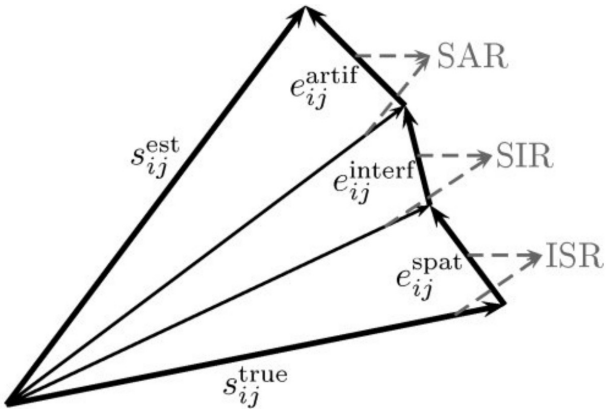


Figure 2.2: Illustration of the relationships of the BSS Eval metrics with i and j denoting the channel and source index, respectively [14].

2.4 Standard pre- and post-processing procedures

The vast majority of methods operate on time-frequency (TF) representations of audio, as music signals in the TF-domain exhibit a clearer structure, sparsity and less overlap. In addition, spectral analysis may be seen as a type of pre-whitening: a time-domain audio waveform has a rather dense covariance, while the covariance matrix of the complex spectrum exhibits an approximately diagonal structure. This pre-whitening effect depends on the chosen frame-length though, because long frames break the stationarity assumption while short frames yield low frequency resolution and thus more correlation. Typically the short-time Fourier transform (STFT) is used, which is a computationally efficient, invertible, and linear operation and hence very well suited to the task. However, other transformations such as the constant-Q transform [15], [16] are also applicable and can be a sensible choice as well.

After estimates $\boldsymbol{Y}_n$ of the sources have been found[1], most approaches take the route of spectral *masking* or *filtering* instead of outputting the estimates directly, and use the estimates to construct masks. The STFT of the source signal may then be recovered by applying its corresponding mask $\boldsymbol{M}_n$ to the STFT of the mixture signal using

$$\hat{\boldsymbol{S}}_n = \boldsymbol{M}_n \odot \boldsymbol{X} \tag{2.12}$$

where $\odot$ represents the element-wise Hadamard product.

This filtering operation may be considered as a form of dynamic equalization, quickly changing with the frame rate. If the masks are constructed appropriately, the estimates are guaranteed to be properly scaled in the sense that they are bounded by the input and their linear sum perfectly reconstruct the mixture signal. There are basically two types of masks: the *binary* mask assigns each TF-bin of the mixture to belong exclusively to one of the sources by either setting it to 0 or 1, and *ratio* or *soft* masks, which may take on any value on the bounded interval $[0, 1]$ and typically produce smoother, more perceptually pleasing estimates that are less interspersed by artifacts (i.e. musical noise). The mask is commonly constructed as a Wiener filter [17], since it turns out to be the minimum mean-squared-error (MMSE) estimator if the sources are assumed to be uncorrelated, zero-centered and wide-sense stationary [18]. Spectral masking as a post-processing step is thus also able to significantly enhance the separation quality. In essence, Wiener filters estimate the true source by first normalizing the model source by the mixture model, before scaling it by the true mixture. In practice, real-valued masks are commonly

[1] Unless otherwise specified, variables in capital letters refer to time-frequency representations.

computed via generalized Wiener filtering (GWF) [19] as

$$\boldsymbol{M}_n = \frac{|\boldsymbol{Y}_n|^\alpha}{\sum_{n=1}^{N} |\boldsymbol{Y}_n|^\alpha} \tag{2.13}$$

since most methods typically operate on magnitude spectrograms only and this generalized formulation includes the classical Wiener filter with $\alpha = 2$. Fractional power spectrograms have been found to better fit the additivity assumption for the mixture model [19]. This general formulation models the underlying stochastic processes as locally stationary and alpha stable distributions instead of Gaussian distributions, which seems to be better suited for audio signals because of their inherently large dynamic range and the resulting large deviation from the mean. Interestingly, the choice of α also appears to influence the perceptual quality of the resulting source estimates, and smaller values of α tend to be more appropriate in the case of music signals [19], [20]. In addition, [21] demonstrates that this spectral magnitude exponent is dominated by the phase information and also argues for a value of α slightly above 1, regardless of the source distribution.

After calculating the mask, the magnitude spectrogram of the source can be obtained. Most methods subsequently simply apply the original phase of the mixture for inverse transformation to the time-domain in order to yield a waveform representation of the source estimate, since phase information is often considered irrelevant [22]. Indeed, utilizing the mixture phase works well in practice and is known to be optimal in the MMSE sense for speech signals [23], however only under certain statistical assumptions which in general do not apply to real-world speech or music signals. Therefore, phase reconstruction has gained a lot of attention from researchers, and is considered to be particularly relevant to perception [24], [25].

2.5 Related work

In the following, a short introduction to the most common approaches to source separation is given. Although this thesis deals with data-driven methods based on deep learning, it may nevertheless be valuable to gain some insights in more traditional model-based concepts in Sections 2.5.1-2.5.3, before turning to the review of some state-of-the-art DNNs in Section 2.5.4.

2.5.1 Pitch-based models

As already stated, the problem of source separation typically suffers from several indeterminacies, that have to be overcome by exploiting intrinsic properties of the mixture and source signals. Model-driven approaches achieve this by explicitly integrating prior knowledge about the nature of the sources into their description of the target signal.

One rather intuitive way of decomposing musical mixtures is to exploit the harmonic structure that is widely present in many sources. Sources associated with lead melodies or also just chords may be explicitly assumed to be of a harmonic nature most of the time. If the fundamental frequency of such a source can be tracked over time and the energy distribution at its integer multiples, its harmonics, can be estimated at each time frame, this description should suffice to either filter out or resynthesize the source. Successful pitch detection is thus a necessary and crucial requirement for such an approach to work. However, this might turn out to be very difficult in the context of polyphonic mixtures due to large frequency overlaps, and multi-pitch estimation alone poses a challenging problem [26]. But also for e.g. lead and accompaniment separation pitch estimation is non-trivial, since the common assumption that the loudest harmonic corresponds to the lead signal does not apply in general, and silences, for instance, have to be handled properly. Also, the lead signal typically includes non-harmonic components - vocals

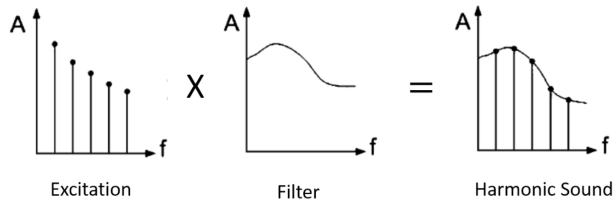


Figure 2.3: The concept of the source-filter model, illustrated in the frequency domain [28].

may be unvoiced, and even whispered or screamed - which break the underlying assumption of the algorithm and have to be accounted for [27]. After estimating which harmonics of the source are present, an estimate of the spectral envelope determines their relative strengths or amplitudes. In a multi-pitch scenario with frequency overlaps, this may include some type of soft-masking in order to distribute the energy of one harmonic present in the mixture properly to the sources. The segregation between excitation and spectral envelope is often accounted for by means of the *source-filter model*, which is especially popular in speech processing. In this case, the voiced excitation signal from the vocal folds is often modeled using a harmonic comb or impulse trains, and noise is used to simulate the airflow of unvoiced sounds. This excitation signal is then convolved with a filter corresponding to the vocal tract, which modifies the slowly varying spectral envelope of the resulting signal according to its frequency response. The source-filter model is depicted in Fig. 2.3.

Pitch and spectral envelope are additive and thus clearly separable in the so-called cepstrum, which therefore is often employed to disentangle their effects [29]. For instance, Mel-frequency cepstral coefficients (MFCCs) are able to represent the spectral envelope, but other yet similar representations might be more useful in the context of music source separation [30]. Spectral envelopes can e.g. also be represented using models obtained by clustering harmonic structure features [31]. In order to generate realistic sounding estimates with sinusoidal models, detailed knowledge about the the target source is necessary, which might be hard to extract from the mixture. Usually pitch-based models include assumptions about the number and type of the sources [32], but there are many ways of inducing priors, e.g. by considering typical melody lines for a particular source [33]. In practice, consistent source estimates are desired, and pitch estimation and timbre modeling will usually have to be considered jointly. For instance, grouping cues such as time-frequency proximity have to be utilized in order to allocate the right pitches to the right sources in a polyphonic scenario.

2.5.2 Spectrogram factorization

Non-negative Matrix Factorization - NMF

Given a 2-dimensional input $\boldsymbol{V}$ that has rows corresponding to the feature dimension and columns containing different data samples (or time steps), factorization methods try to find a good approximation

$$\boldsymbol{V} \approx \hat{\boldsymbol{V}} = \boldsymbol{W}\,\boldsymbol{H} \tag{2.14}$$

by decomposing it into a *dictionary* $\boldsymbol{W}$ of recurring patterns that are characteristic for the data, and a matrix $\boldsymbol{H}$ whose columns hold so-called *activations* which determine the relative composition of the dictionary's building blocks for the approximation of each column of $\boldsymbol{V}$. The number of dictionary entries (which is equal to the common dimension of $\boldsymbol{W}$ and $\boldsymbol{H}$), is often referred to as the factorization's rank K. In the case of audio source separation, $\boldsymbol{V}$ usually corresponds to the mixture's magnitude or power spectrogram. After an approximation has been found, the source may subsequently be identified by clustering (in an unsupervised setting)

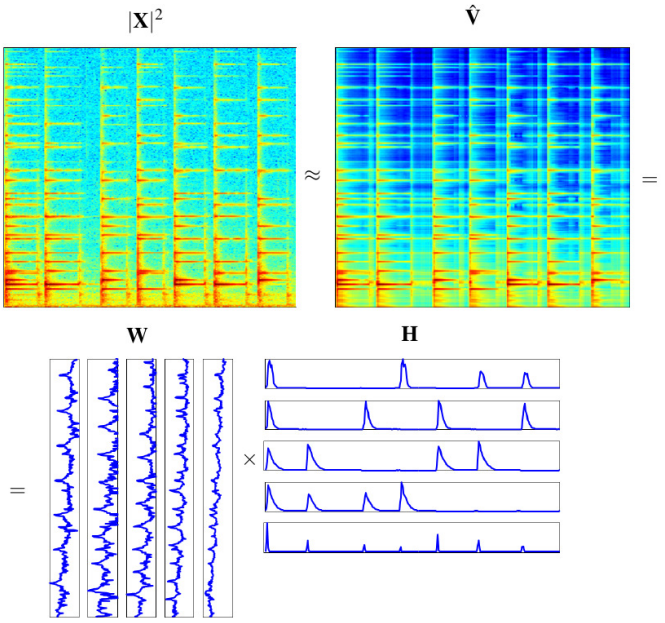


Figure 2.4: Power spectrogram of a short piano sequence composed of four notes and its corresponding low-rank approximation obtained by applying non-negative matrix factorization with rank $K = 5$ [38].

or selecting (supervised setting) the entries in the dictionary that correspond to the target source. There are various ways to obtain such a factorization, for example by applying the well-known principal component analysis (PCA) and optimizing the approximation with regard to the minimum mean square error [34]. Independent component analysis (ICA) is a related method that estimates an unmixing matrix based on the statistical independence of the sources [35]. However, ICA is a rather unreliable approach, because music sources are highly correlated both in time (common onset / offset times) and frequency (e.g. due to overlap of harmonics). Arguably the most popular low-rank approximation method for blind source separation is *non-negative matrix factorization* (NMF). In contrast to simple PCA which allows real-valued decomposition components and produces holistic representations well-suited for compressing the entire given data, NMF constrains optimization to non-negative values for $\boldsymbol{W}$ and $\boldsymbol{H}$, ensuring that the obtained dictionary entries are additive and *parts-based* building blocks [36], [37]. This turns out to be very useful in the context of audio signal separation, yielding an elegant, flexible and most importantly interpretable framework for source modeling. Since the input $\boldsymbol{V}$ will usually be non-negative as well, methods based on NMF may be very straightforwardly applied to the mixture magnitude or power spectrogram. The concept is illustrated in Fig. 2.4.

Measures of fit for the optimization of NMF based algorithms for audio signals are typically part of the family of β-divergences, which include the Itakuro-Saito divergence ($\beta = 0$), the Kullback-Leibler divergence ($\beta = 1$) as well as the quadratic cost ($\beta = 2$) as special cases [39], and also allow for a smooth interpolation between these cases by adjusting the value of β. The choice of the objective function indeed has a significant impact on the approximation's behavior: while the Itakuro-Saito divergence precisely models lower energy (usually high-frequency) TF-bins, the Kullback-Leibler divergence relies more on large data values (usually lower-frequency TF-bins). In fact, considering an equivalent probabilistic formulation of NMF [40], both cost functions respectively act as the optimum estimator in the maximum-likelihood sense for different underlying probabilistic assumptions [41]. Using a value of $\beta = 0.5$ appears to be a sensible trade-off in the case of music signals [42]. Most non-negative matrix factorizations are optimized using the Majorization-Minimization algorithm, which iteratively proceeds by minimizing a convex auxiliary function that forms an upper bound of the original cost for the current estimate.

It turns out that the decomposition of a β-divergence into a sum of a convex and a concave component allows to construct a majorized auxiliary function that has a closed-form minimizer [39]. Following a block-coordinate descent procedure, the resulting simple multiplicative updates can then be applied alternatingly for $\boldsymbol{W}$ and $\boldsymbol{H}$ by assuming the respective other component to be fixed. Since $\boldsymbol{W}$ and $\boldsymbol{H}$ are interchangeable by transposition $\boldsymbol{V}^T \approx \boldsymbol{H}^T \cdot \boldsymbol{W}^T$, the updates are essentially the same for both components. Note that this algorithm strongly depends on initialization and is not guaranteed to converge to a global (or even local) minimum.

After the optimization problem has been solved, unsupervised source separation via NMF can be accomplished by clustering dictionary components or analyzing activation events e.g. according to their duration. However, real-world audio signals exhibit highly complex behavior and comprise a large number of varying components, and vanilla unsupervised NMF without integration of prior information only works well on toy-data. However, it is very straightforward to employ NMF in a supervised setting, by learning a dictionary $\boldsymbol{W}$ from training data, discarding its accompanying trained activation matrix $\boldsymbol{H}$, and re-estimating $\boldsymbol{H}$ from the test signal with the fixed dictionary [43]. For source separation in particular, several source-specific dictionaries $\boldsymbol{W}_n$ may be learned from clean signals for N different sources and concatenated into one large matrix $\boldsymbol{W} = [\boldsymbol{W}_1 \cdots \boldsymbol{W}_N]$. Upon factorization of the test signal, the source estimates $\boldsymbol{Y}_n$ may then be clearly determined by extracting the associated fixed dictionary entries $\boldsymbol{W}_n$ along with their corresponding (re-estimated) activations $\boldsymbol{H}_n$, for instance via Wiener-filtering:

$$\boldsymbol{Y}_n = \frac{\boldsymbol{W_n}\,\boldsymbol{H_n}}{\boldsymbol{W}\,\boldsymbol{H}} \odot \boldsymbol{V} \tag{2.15}$$

Semi-supervised approaches proceed in a very similar way, the main difference being that only a subset of the source dictionaries is assumed to be fixed while the remaining ones are estimated from the test signal directly [44]. Alternatively, instead of computing dictionaries, the training data itself might be used $\boldsymbol{W}_n = \boldsymbol{V}_n$ as the dictionary matrix (if it is not too large), or clustered and averaged versions of its spectral samples / columns [45].

A variety of extensions to NMF algorithms with applications to audio source separation have been proposed in the literature. For instance, instead of incorporating only one spectral component per column in the dictionary, one dictionary entry might contain a patch of components each of which corresponds to the same single activation row-vector $\boldsymbol{h}_k$. A single column or time frame of the approximation then computes as $\boldsymbol{v}_t = \sum_{k=1}^{K} \sum_{l=1}^{L} \boldsymbol{w}_{kl} h_{kt}$, which can be referred to as a convolutive approach to NMF [46]. Similarly, it is possible to enforce certain structures that are characteristic of musical sources by representing dictionary elements, for instance, as a weighted sum of narrowband spectral patterns (either corresponding to narrowband noise or pitched harmonic series), or by reproducing the source-filter model as a product of excitation and filter spectra, which might bring along favorable conditions to tackle singing-voice separation [47]. These types of modifications can be conveniently described in more general terms by extending NMF to multi-dimensional arrays [48], [49]. A different way of incorporating prior knowledge is by explicitly inducing sparsity into the factorizing components [50], [51]. This is especially appropriate when speech or vocals are to be separated, as these sounds typically feature less active frequency bins, less redundancy and complex harmonic structure. Also, NMF with a large rank K but without additional constraints can lead to trivial, indiscriminate spectral bases. Encouraging sparsity may be able to alleviate this issue. In particular, $\boldsymbol{H}$ can be forced to only activate few dictionary entries at a time by adding a penalty term to the objective function. Typically the ℓ_1-norm $\|H\|_1$ is chosen, but different sparsity promoting penalties exist. In a similar manner, sparsity may be induced in terms of groups, for instance in order to simultaneously activate several dictionary elements that jointly account for a certain phoneme, note or source [52]. Another type of NMF is the so-called projective NMF, which has been used to explicitly capture the musical concepts of harmonicity and percussivity [53]. Finally, NMF based algorithms may take into account the temporal dynamics of a source [54], [55]. For ex-

ample, a penalty for largely dissimilar neighboring activations may be added, ensuring temporal smoothness which is useful for overly harmonic sources. Similarly, such dynamic models are able to factor in time-varying activations, e.g. with regard to typical attack-time envelopes.

Robust Principal Component Analysis - RPCA

As stated above, most TF-bins of music mixtures may be assumed to belong to the accompaniment which exhibits repeating structures and recurring similar sounds, while vocals are sparse and show high variation. Robust principal component analysis (RPCA) is a method somewhat related to NMF, which makes use of these properties and assumes that the mixture is a composition of underlying sparse (e.g. vocals) and low-rank (e.g. accompaniment) components. It solves the convex optimization problem

$$\text{minimize} \quad \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{V} \tag{2.16}$$

where $\|\cdot\|_*$ denotes the nuclear norm (the sum of singular values) which is often employed to find low-rank matrices, and $\lambda > 0$ corresponds to a trade-off parameter between the rank of $\boldsymbol{L}$ and the sparsity of $\boldsymbol{S}$. RPCA has been successfully applied to singing-voice separation [56]. Analogous to NMF, this approach is highly interpretable and hence lends itself to a variety of extensions, for instance by introducing harmonic and regularization constraints as in [57]. Music structure analysis such as vocal activity detection prior to source separation has also proven to boost performance and is in general very helpful for methods based on spectrogram factorization [58].

2.5.3 Kernel-based models

Repeating Pattern Extraction Technique - REPET

While the previous sections discussed methods that explicitly parameterize the target source's power spectral density, a different group of non-parametric methods exploits only local regularities, which may be summarized as kernel-based models. A notable representative of these methods is the so-called repeating pattern extraction technique (REPET) [59]. REPET is used for foreground/background separation and is based on the simple assumption that the musical background has an underlying repeating structure and is thus more redundant than the overlying foreground (e.g. vocals or lead instrument). To this effect, it should be possible to infer a mask that extracts the background sources by distinguishing between roughly periodically repeating and non-repeating TF-bins. The technique bears resemblance to background subtraction methods in computer vision. The concept of the algorithm is depicted in Fig. 2.5 and in general proceeds as follows: after analyzing the given musical mixture with regard to repeating structures, for instance by computing a beat spectrum [60], these repeating patterns can be identified in the input mixture and combined to yield a smooth representation of the repeating segment via median filtering. This segment may then be used to construct a TF-mask for background extraction.

The degrees of freedom of this method mainly consist in the different ways of identifying repeating patterns. For instance, to account for varying structures (such as verse and chorus of a song), one might apply REPET on individual windowed and possibly overlapped segments in order to extract the local repeating background, or use an adaptive version that computes a beat spectrogram instead of a global beat spectrum. Another extension, REPET-SIM, generalizes the previous approaches and takes into account repeating yet non-periodic patterns by utilizing a similarity matrix to identify these repeating patterns. Additionally, variants of REPET well-suited for on-line processing do exist.

Harmonic/Percussive Separation using Median Filtering

A different kind of kernel-based method deals with the separation of harmonic and percussive components of musical mixtures [61]. The idea behind this approach is simple yet extremely effective: since percussive and harmonic sounds are highly localized in time and frequency, respectively, they can be conveniently separated using median filters across successive frames (for harmonic components) and across successive frequency bins (for percussive events).

Kernel Additive Models - KAM

Both of these methods regard interference from unwanted sources as outliers, and rely on median filtering as an estimator which is robust with regard to these outliers. Similarly, both algorithms exploit local features which are observable in the mixture spectrogram. The framework of Kernel Additive Modelling (KAM) generalizes these methods by introducing the notion of proximity (in terms of e.g. repetition, continuity, stability, self-similarity, common fate, etc.), and proposes that a certain TF-bin associated with the target source may be determined from other TF-bins of the mixture by applying a so-called proximity kernel [62]. REPET-style algorithms and the harmonic/percussive median filtering algorithm thus are versions of KAM for different choices of the proximity kernel, as illustrated in Fig. 2.6. Additionally, [62] introduces a method to find suitable kernels using the iterative kernel-backfitting algorithm. The KAM framework has been subject to modifications too, for example with regard to its efficiency [63] and the kernel-backfitting optimization procedure [64].

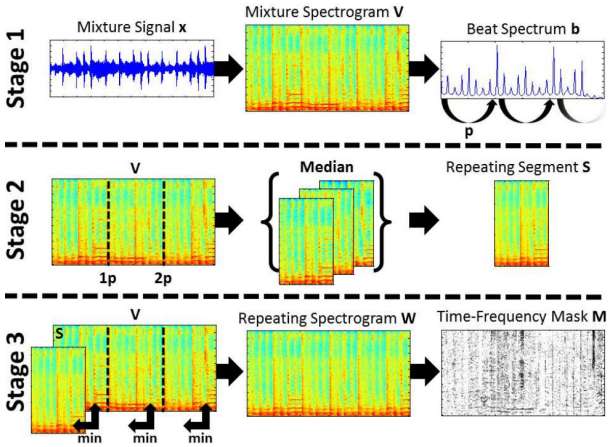


Figure 2.5: Visualizazion of the original REPET approach for foreground/background separation [59].

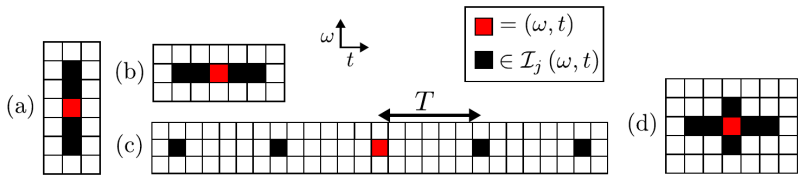


Figure 2.6: Different proximity kernels of KAM-methods [62]. w denotes the frequency-axis, t the time-axis. Red bins correspond to the current TF-bin in question for a particular target source j. Black bins are part of the neighborhood or proximity I_j, which are TF-bins for the source j that are assumed to be approximately constant in the mixture spectrogram. In this illustration, (a) and (b) represent proximity kernels of the median harmonic/percussive filtering algorithm, (c) is associated with REPET-style algorithms, and (d) might correspond to a kernel for vocal separation since it operates both on the time- and frequency-axis.

Of course, different ideas from all of the approaches to source separation that have been discussed may be combined in order to obtain a more robust system, and many examples can be found in the literature. To mention a few examples, NMF and RPCA may be combined with pitch-based methods in order to ignore or weigh TF-bins in the mixture that correspond to the harmonics of a lead melody, such that mainly the accompaniment is subject to factorization [65], [66], whereas [67] uses a pitch-based method in combination with the REPET-SIM approach. In a similar manner, it is possible to cascade multiple algorithms [68], [69], or yield the final result from an estimate of an ensemble of different methods [70].

2.5.4 Deep Neural Networks

Although many of the aforementioned methods are also suitable for supervised training settings, they enforce certain desired properties on the source signals, which makes them prone to failure if the sources do not exhibit these properties or do not obey their signal model. The problem of source separation has seen dramatic improvements in performance since the advent of modern deep learning, as did many other fields of research in signal processing and computer science in general. Despite the fact that DNNs are usually referred to as *models* as well, they do not introduce assumptions on the signals and are less restrictive in the sense that they act as universal function approximators [71], [72]. If enough representative training data is available to train the network's parameters on the minimization of some reconstruction loss, the need to explicitly model the spectral characteristics of a source is eliminated.

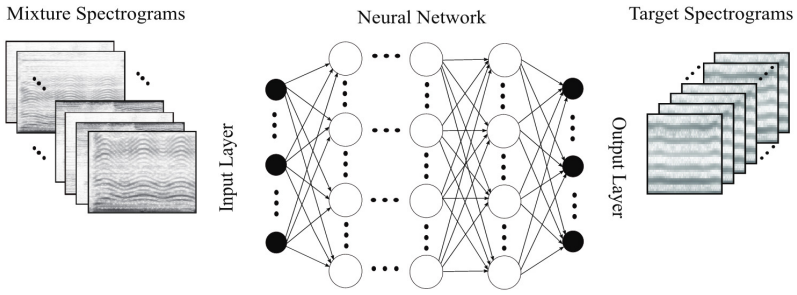


Figure 2.7: Illustration of the basic concept of source separation in the time-frequency domain using a simple feed-forward DNN [73]. In this case spectrograms are inferred directly, but mask estimation is possible and common as well.

Most source separation algorithms based on deep learning take the mixture signal as input and output the target source signal, as shown in Fig. 2.7, or a corresponding mask. Usually one model per source is employed for the decomposition of a given mixture. This is reasonable with regard to the efficiency objective of this thesis, as compute power is not unnecessarily spent on estimating extraneous sources. However, estimating only one source spectrogram or mask prevents utilization of the well-established and indeed highly beneficial post-processing step of Wiener filtering, which requires an estimate of the entire mixture model in the denominator. Interestingly, the supposedly first application of DNNs to music source separation addresses this problem, by jointly computing representations of all sources and using fixed deterministic layers to directly infer masks from the network [74]. Another contribution of this work is the usage of recurrent hidden layers, which allows for an extremely reduced context size of 3 frames. After investigating the use of fully-connected feed-forward neural networks (FNNs) for music source separation in [75], [76] showed that the use of bi-directional long short-term memory networks (Bi-LSTMs) are better suited to the task, and that an ensemble of both methods consistently produces improved estimates. Moreover, [76] also demonstrates the effectiveness of data augmentation techniques: although mixing sources from different parts of different songs

and applying random gains generates incoherent and unrealistic mixtures, the expanded amount of training data (which is particularly scarce in the case of music signals due to copyright protection) helps the network to generalize better.

In general, encoder-decoder topologies dominate the literature of deep learning for music source separation. The encoder transforms the input mixture to a compact representation that is useful for separation of the target source, and the decoder generates the desired target from this encoding. An encoder-decoder architecture based on gated recurrent units has been proposed in [77] and has seen extensions in [78] and [79], which feature a denoising highway-network after the encoder-decoder masking network for artifact reduction. However, the recurrent inference, the parameter-heavy model and the intricate training procedure paired with the moderate separation performance compared to other state-of-the-art methods, let this approach appear less attractive. Another, more recent model that also relies on recurrent layers is called open-unmix (UMX) [80], and employs 3 hidden Bi-LSTM layers as well as fully-connected layers at the input and output. While the main intention of the authors was to provide a baseline model with an open-source reference implementation, UMX achieves state-of-the-art separation quality. Instead of RNNs, systems based on convolutional neural networks (CNNs) can also be very successfully applied to the problem of music source separation (e.g. [81], [82], [83], [84]), prevalently in the form of *U-Nets*, which will be discussed in the following. Despite the stateless system architecture, crucial temporal dependencies in the audio material may still be exploited by simply aggregating contextual information and passing this "context data window" as input to the network. [85] even advocates the use of CNNs in the context of music source separation as opposed to RNNs. Unsurprisingly though, a combination of the strengths of both approaches yields even higher separation quality [86], [87], by extracting abstract local features via convolutional filters and capturing long-term structure through subsequent recurrent layers. At this point it should be noted that architectures based on attention-mechanisms [88] have recently been applied to the task of music source separation and showed state-of-the-art results [89], [90], however, these methods are not discussed any further in the context of this thesis.

U-Nets

U-Net structures [91] have proven to be particularly effective and have become arguably the most used CNN configuration in audio source separation and similar regression tasks [81]. The rationale behind the U-Net topology depicted in Fig. 2.8 is that the network does not have to operate entirely at the full resolution. Instead activations may be downsampled or *pooled* at several intermediate stages and subsequently upsampled accordingly, e.g. via *transposed convolutions*, to recover the target scale. While capturing and preserving fine detail and local information at higher resolutions, layers at lower scales can benefit from an extended spatio-temporal context as well as large receptive fields and may learn to produce more general, global and abstract features. Note that this architecture resembles typical denoising auto-encoders, and the problem of audio source separation may indeed be reformulated as the task of *denoising* a corrupted version of the desired target source. Additional skip-connections between encoding and decoding layers of the same scale allow to keep both kinds of information (local and global) at once. The effect of skip-connections has been studied in depth in [92], and has been found to encourage auto-encoder-like networks to learn non-trivial mapping functions, especially if the network's input and output are skip-connected. Additionally, residual- or skip-connections are known to tremendously help the optimization procedure since they alleviate the vanishing/exploding gradient problem [93], [94]. Many successful source separation systems build on the U-Net architecture, such as [83], [84], [86], [95], [96] and [87].

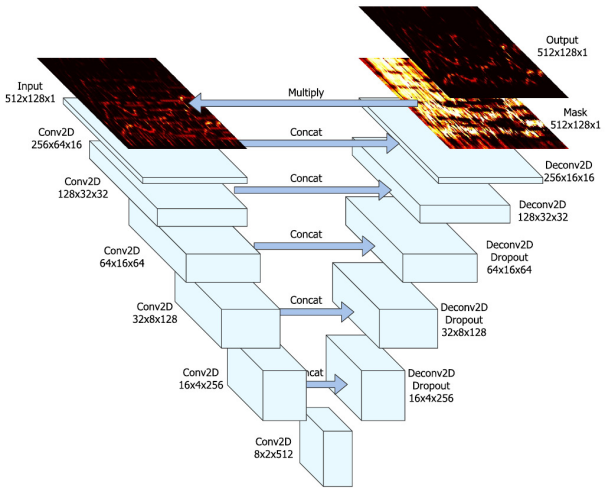


Figure 2.8: Network architecture of the U-Net used for singing-voice separation [81].

Time-domain DNNs

Since the introduction of the audio synthesis model Wavenet [97], methods that are capable of operating in the waveform-domain have become increasingly popular due to their inherent ability of exploiting and modeling phase information, and the on-going pursuit of truly *end-to-end*-trainable models. Instead of explicitly imposing a time-frequency representation, time-domain DNNs learn task-specific representations using their own set of basis functions. However, the performance of time-domain models for music source separation such as [98], which builds on the Wavenet, and the Wave-U-Net [96], albeit yielding decent results, initially lagged behind methods that relied on explicit spectral analysis. This changed with the introduction of Conv-TasNet, a time-domain model originally proposed for speech separation [99] that obtains state-of-the-art separation performance on various types of sounds [100] including music signals: [87] uses an adapted Conv-TasNet as a baseline model, points out potential shortcomings of the architecture and proposes a different U-Net-style waveform model called Demucs, which produces similar though slightly more perceptually pleasing results. [101] conducted an in-depth empirical study of the architecture and provides insights into its generalization capabilities which also apply to music source separation. Recent work conducted in [102] also applied and extended Conv-TasNet for music source separation. The authors used a meta-learning approach by employing a masking extractor specific to each source and a parameter generator that predicts the parameters of the extractor conditioned on a one-hot encoded input vector of the source. In doing so, they also address interesting issues such as parameter-efficiency, multi-samplerate stages and the use of the STFT as an additional parallel feature extractor to the learned basis functions of the 1D-convolutional kernels.

Training objectives

Apart from finding appropriate network architectures, the deep learning literature specific to audio source separation deals with exploring different training procedures and, in particular, objective functions. As far as models in the TF-domain are concerned, most methods simply take the mean squared error between the estimate and the target as a measure of fit. Despite being simple and effective, the use of the Euclidian distance is not very well motivated from a perceptual viewpoint, as TF-bins do not contribute equally to a certain sonic sensation. We are rather interested in correctly learning the "distribution" of the target's TF-bins. Therefore divergences such as the Kullback-Leibler (KL) divergence seem like a more reasonable and appealing choice [103], [104]. However, the resulting performance appears to be model-dependent. Simple MSE

loss functions tend to be more popular because they work very well in practice, and are also used by recent state-of-the-art methods (e.g. [86], [89]). A different loss function for spectral speech estimates that essentially optimizes the signal-to-noise ratio (SNR) has been proposed in [105]. Another promising idea for measuring the quality of the estimates is to compute the pixel-wise difference to the target in a high-level feature space instead of the TF-domain. This can readily be achieved by feeding the spectrograms (or masks) to another DNN, which may have been trained on e.g. audio or even image classification tasks, and computing a loss term from the features extracted by one or more hidden layers of this network. [106] investigated this approach and showed that it consistently improves the separation performance for music signals. In the case of time-domain methods, it is natural to train the end-to-end model directly on performance based cost functions, such as the SDR or the scale-invariant SDR (SI-SDR) [107]. However, the ℓ_1-loss is an equally valid objective function for time-domain DNNs and has also been utilized to yield state-of-the-art performance, as demonstrated by [87].

A different kind of training objective is offered by the *deep clustering* framework [108], which aims at generating high-dimensional embeddings that allow for convenient clustering of TF-bins with respect to the sources. Deep clustering has been originally proposed in order to facilitate speech source separation and the accompanying permutation problem, and has since been extensively used for this purpose (see e.g. [109] for a recent state-of-the art speech source separation system), but can also be applied to music source separation [110]. Section 4.6 discusses this type of loss function in greater detail.

Training data

Neural networks reduce the need for rigorous constraints on the signal model by directly learning from training examples and in doing so pushed the boundaries of source separation performance. However, this data-driven approach can pose potential shortcomings if the DNN is not able to properly generalize due to a lack of training examples. Compared to speech separation, "labeled" multi-track music mixture data with clean source signals is particularly scarce due to copyright protection, and previous datasets such as *iKala* [111], *ccmixter* [112] and *medleydb* [113] are deemed to be insufficient. This circumstance is also evident from the timeline of successes in the field of deep learning based music source separation, which moved forward in leaps alongside the release of adequate datasets such as *DSD100* [114] and its extension *musdb18* [115], which has become the most popular and widely used dataset for music source separation. The recently introduced database *slakh* [116], which is based on rendered MIDI data, will certainly further contribute to this development. Furthermore, effectively utilizing deep learning despite little available training data is an active field of research. The most common way of dealing with this issue is by artificially expanding the available dataset via augmentation methods that generate new training examples by applying some kind of transformation on the original data [76], [117]. Data augmentation techniques have become standard components of DNN training procedures, and are also subject to research that tries to better understand the mechanisms behind their effectiveness [118]. Another strategy, termed *transfer learning*, transfers a network that has been pre-trained on a large dataset from a different but related domain to the target domain by re-training it on the smaller in-domain data [119], [120]. Additionally, [121] states that deep neural networks are indeed capable of generalizing on smaller datasets as well.

Resource efficiency

There is a great demand for the deployment of deep learning based systems in resource-constrained environments in general. Transferring DNNs from scientific and cloud-based environments with virtually unlimited computing power to consumer-grade electronic hardware for on-device inference is among the major challenges of modern machine learning. Additional real-time require-

ments make this task even more difficult. Solutions to this problem are required to find a sensible trade-off between the model's performance and its efficiency, which is determined by its memory footprint, inference speed and energy efficiency. A comprehensive overview about resource-efficient approaches to deep learning is given in [122], [123]. These can essentially be categorized into 3 different classes: quantization methods, that try to represent weights and/or activations using less precision than the typically used 32-bit floating point values; pruning methods, which remove parts of the network architecture during or after training; and approaches that strive to improve the network's efficiency at a structural level, for instance via weight sharing, special matrix structures and lightweight building blocks, knowledge distillation and neural architecture search. The literature concerning efficient deep learning based music source separation is rather limited. One convolutional method which focuses on low latency has been proposed in [124]. It adheres to the encoder-decoder approach, however, it uses several decoders in order to simultaneously estimate multiple sources. While the work conducted in [124] is of special relevance to this thesis, the slimness of the model paired with the shared representation of multiple sources limits the expressive power of the model. Strategies to meet efficiency requirements will be dealt with in greater detail in Chapter 5.

3

Baseline system and experimental setup

This chapter describes the foundations for the experiments, before the results are presented in the following chapters. First, Section 3.1 considers relevant properties that an appropriate baseline system with regard to a balanced trade-off between performance and efficiency should comply with, and justifies the choice of the state-of-the-art MMDenseNet architecture, which will be explained in greater detail in Section 3.2. Details regarding the experimental setup, which relate to the used dataset, data pre-processing, the optimization procedure and additional hyperparameters, are presented in Section 3.3.

3.1 Determining a baseline

Determining an appropriate baseline system for this work not only depends on its capability of producing high quality separation estimates. Since the same model should also work as a foundation for considerations towards efficiency, it is of equal importance whether the model obtains these results in an inherently resource-friendly, computationally effective and fast manner.

Recurrent vs. convolutional units

While the use of recurrent neural networks (RNNs), and Bi-LSTMs in particular, may seem like a suitable choice given the sequential nature of audio data, they are computationally expensive and parameter heavy - even more so as they are often used in conjunction with fully connected layers. In addition, switching from bi-directional to causal units for utilization in on-line settings may lead to serious performance degradations. CNNs on the other hand are much more lightweight (e.g. due to shared weights) and have a clear advantage over RNNs in terms of efficiency. In contrast to sequential RNNs, the stateless kernel-based processing of CNNs allows for much better utilization of parallel hardware, which is available in virtually any of today's computing devices and thus encourages the use of CNNs in resource-constrained environments. In addition, recent research towards the development of more efficient DNNs mainly concentrates on convolutional architectures, due to their ubiquitous presence within image processing applications (see Section 2.5.4 and Section 5.4). As outlined in Section 2.5.4, hybrid models are able to combine the strengths of both topologies. However, the main concern of this thesis is the efficiency of source separation systems based on deep learning, since it is not clear whether it is at all possible to implement a modern large-scale DNN given the rigorous real-time requirements of small, real-world embedded systems with limited computing powers. Additionally, the incorporation of recurrent units into hybrid architectures is considered to be rather straightforward (cf. [84], [86]), and may be added after the feasibility of a certain fully convolutional model has been confirmed in this context.

Time vs. time-frequency domain

DNNs operating in the waveform-domain have advanced to the field of music source separation by now and indeed generate impressive results. However, a few aspects of time-domain models

raise some considerations with regard to their appropriateness for the objective of this thesis. First of all, the algorithm for calculating the fast Fourier transform (FFT) is highly optimized and requires only negligible computation cost on general purpose processors. In contrast, transforming the input from the time-domain to the learned representation of the network is likely to be significantly less computationally effective, while the transformation might be very similar to that of a plain STFT. And although incorporating phase information as well as employing a task-specific representation appears to be highly beneficial, there are some reasons to doubt the effectiveness of learned basis functions. For instance, [100] used the Conv-TasNet architecture to compare the performance of learned features with that obtained using STFTs, and discovered that the latter works better in certain cases. The Meta-Tasnet [102] extracts features using both learned basis functions as well as an STFT, but the relative contribution of each of the transformations to the result is not clear from the paper. Furthermore, very recent attention-based neural networks [89], [90] currently outperform all other state-of-the-art methods, while estimating magnitude spectrograms purely in the TF-domain and simply using the original mixture phase for inverse transformation. For these reasons, this thesis focuses on fully convolutional neural networks that operate in the TF-domain, in particular the *MMDenseNet* introduced by Takahashi and Mitsufuji [84].

3.2 Multi-scale Multi-band DenseNets

DenseNets

The increase in performance as well as popularity of neural networks, in particular CNNs, roughly correlates with the increase of their depth [125], [126]. Presumably, the advantage of deeper models lies in their capability of modeling hierarchies of increasingly complex features or concepts. However, the propagation of information both backward and forward is impeded as networks grow in depth and managing the training of deeper models is difficult due to problems like vanishing or exploding gradients. In fact, many popular strategies for training DNNs tackle this problem in different ways, e.g. via better initialization, alternate optimizers, pre-training schemes or well designed activation functions.

Another effective and groundbreaking method that enabled the training of extremely deep CNNs is to employ residual- or *skip-connections* that act as shortcut identity functions between layers, which improves information flow inside the network and explicitly allows to learn residual instead of direct mappings [127], [128]. One variant of such an approach is the well known *DenseNet* [129], originally proposed for image classification tasks: the basic idea behind DenseNets is to construct the input to one layer by concatenating (in the channel direction) the feature maps produced by all preceding layers, and the corresponding output activations of the layer are again fed to all subsequent layers, as illustrated in Fig. 3.1.

This "dense" connectivity not only avoids the vanishing gradient problem but also encourages the reuse of features generated by previous layers and thus ensures parameter efficiency. This architecture attains great results, with the only disadvantage being the rather large memory consumption of DenseNets, since the number of feature maps each layer has to process increases linearly in proportion to the number of kernels per layer (which for this reason is also called the *growth rate* parameter, usually denoted by k). The intuition behind utilizing densely connected CNNs for audio source separation is that source spectrograms buried in interference signals can be recovered more easily by frequently referring back to the mixture and intermediate representations, hence exploiting the feature reuse property of DenseNets.

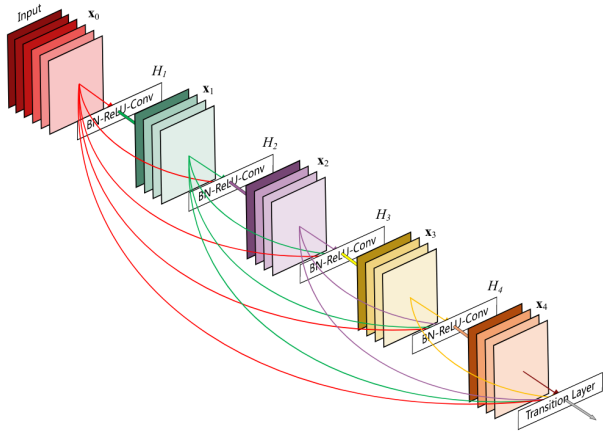


Figure 3.1: The concept of dense connectivity as used in the original DenseNet paper [128].

Multi-scaled DenseNets

In order to adapt the idea of DenseNets to regression tasks, the desired time-frequency resolution (usually matches that of the input spectrogram) has to be maintained at the output. However, simply stacking multiple densely connected layers results in infeasible memory requirements, as outlined above. Instead, [84] adopts the popular U-Net structure described in Section 2.5.4, which explains the "multi-scale" attribute. The authors also validate the U-Net structure by investigating the ℓ^2 norm of weights at different scales, corresponding to either the upsampling or the skip-connection path, which turn out to exhibit a quite balanced distribution and therefore should roughly contribute equally to the result (as depicted in Figure 5 in [84]). The parameter s indicates the number of different scales used throughout the network architecture.

Multi-band processing

Music extends over a broad range of frequencies and different spectral areas exhibit distinct characteristics. Since the training of convolutional kernels will be dominated by signal components of higher energy, this may lead to a poor representation of high frequency content which is typically less prominent. The MMDenseNet tries to compensate for that by employing several networks in parallel dedicated to different frequency bands. The separate outputs can then be concatenated in the frequency direction to retain the full resolution feature maps. This estimate is combined (via concatenation along the channel dimension) with the output of an additional network that covers the entire spectral range, to account for both local and global patterns. In addition to reducing and clearing up the search space for the separate convolutional filters, this method allows for a non-uniform distribution of computational power across frequency bands of varying significance (by adjusting the individual network sizes) and therefore benefits the overall computational efficiency.

Summary of the MMDenseNet architecture

In summary, the MMDenseNet uses three U-Net-like networks assigned to different frequency regions, with one network covering the full spectral range and a higher and lower band associated with the other two (spectrogram is simply split in half at the center of its vertical axis). Following an initial input convolution, the basic processing unit between pooling and upsampling layers of each of those U-Nets is the so-called *dense block*, which itself consists of L densely connected layers. These layers, in turn, are essentially pre-activation convolutions - a composite function made up of a sequence of 2D Batch Normalization ($BatchNorm$), a Rectified Linear Unit ($ReLU$) and a convolutional unit employing k (corresponding to the growth rate of the respective dense

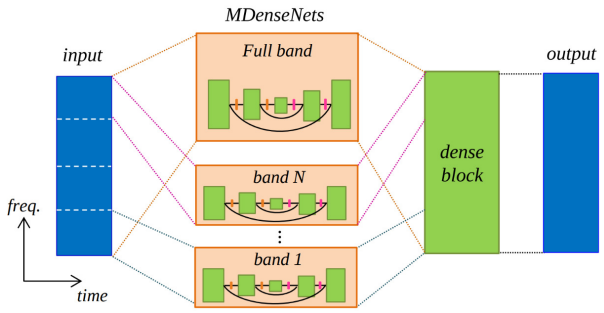


Figure 3.2: Model architecture of the MMDenseNet [84]. Each green block consists of densely connected composite function units composed of BN, ReLU and a 2D convolutional layer. These are arranged in a U-Net-like fashion, forming a single MDenseNet. The MMDenseNet is then made up by operating several MDenseNets in parallel on different frequency regions, plus one dense block in order to gather the individual parallel outputs.

block) 2D kernels of size 3x3. The downsampling layer is defined by a 1x1 convolutional operation (preserving the number of feature maps) followed by an average pooling layer of size 2x2. The upsampling layers consist of 2D transposed convolutions, with the filter size corresponding to that of the pooling operation. After the outputs of all separate networks are gathered and combined, they are passed through a final dense block in order to yield a consistent output. The model architecture is illustrated in Fig. 3.2.

Each source is modeled individually by one MMDenseNet. The final estimates of the original paper are obtained by applying a multi-channel Wiener Filter (see Section 2.4 and e.g. [19]), which requires the availability of all source estimates at once. Although this post-processing scheme considerably enhances the quality of the results, this condition can hardly be met considering that a single model already poses difficulties for real-time scenarios with limited computational resources. Instead, the remainder of this work will neglect the post-processing step and directly takes the output of one MMDenseNet as its corresponding magnitude spectrogram source estimate. This spectrogram may then be transformed to the time-domain using the inverse STFT with the original mixture phase.

3.2.1 Implementation details

All parameters of the MMDenseNet, including the growth rate k, the number of layers L and the scale parameter s are given in Table 1 of the reference paper [84]. A few open questions remain unanswered though, since the implementation has not been made publicly available. This section gives a concise overview about the choices made during the implementation of the baseline system within the *PyTorch* framework [130].

Bias

One detail about the model that is not addressed directly is whether the convolutional layers of the MMDenseNet include bias terms. These are usually omitted by large CNNs which typically are interspersed by Batch Normalization layers and naturally incorporate bias terms that way. We therefore follow this proceeding. However, it is crucial to include a bias term at the output convolution, since failing to do so would unnecessarily condition the previous parameters on the scale of the target output. A similar point can be made for the input layer: although the original DenseNet for the ImageNet dataset employs BatchNorm as its very first operation, this seems not to be the case for the MMDenseNet. For this reason it is wise to decouple the subsequent layers to some extent by including a bias term in the input convolution.

Dense Block Output

Arguably the key component of DenseNets is the concatenation of feature maps across blocks and the unobstructed propagation of data: the output of one composite function, also called *dense layer*, is concatenated with all previous dense layer outputs inside the same dense block, plus the initial input to the block. The output of one dense block, as proposed in the original DenseNet paper [129], is constructed analogously - it is the concatenation of all feature maps that were generated inside it with its input. In contrast, though not clearly stated in text, Figure 1 of [84] indicates that the output of a dense block of the baseline system simply corresponds to the output of the last dense layer of the block. Thus all activations are squeezed through the last composite function, with the resulting output channel dimension equaling the growth rate. As the original version would come with an immense increase of computational cost, our implementation adopts the latter, modified approach concerning dense block outputs.

Multi-band concatenation

Before going into the final dense block, the activations of the independent network components associated with different frequency bands have to be joined. But since different growth rates k are used in each case the output feature map dimensions will not align properly for simple concatenation. In particular, this concerns the features corresponding to the higher and lower frequency regions, since their combination has to yield the full frequency resolution before being concatenated along the channel axis with the output of the full-band network.

While there are several different ways to do this, the authors do not give details about their implementation concerning this matter. One may synchronize the number of kernels used in the final dense layer of the last parallel dense blocks. However, since the disjoint network parts are not equally relevant as outlined previously, this procedure requires a tradeoff between an unnecessary increase in computational inefficiency and a negative impact on the model's capacity to represent a particular band. Another method of handling this issue would be to add a subsequent 1x1 convolution and map all outputs to the same feature map dimension. While somewhat compensating for differing feature representations, this approach is subject to a similar tradeoff problem and additionally hurts efficiency by introducing a new convolutional layer. The most straightforward way, however, is to simply zero-pad the missing dimensions. Though lacking an explicit alignment of the different feature representations, it has to be learned by the network anyway. Furthermore, this method is efficient, avoids the outlined tradeoff and is therefore adopted for this implementation.

3.3 Experimental setup

3.3.1 Dataset

All experiments presented in this thesis have been conducted using the *musdb18* dataset [13], which consists of roughly 10 hours of western pop music with stems for vocals, drums, bass sources as well as one source that groups all other occurring sources. It comprises 150 full-length tracks in total and has dedicated train and test sets composed of 100 and 50 songs, respectively. All songs are stereophonic and encoded at 44.1 kHz. A validation set has been determined by randomly selecting 20 of the 100 songs in the training set. To avoid over-representing longer tracks during training, we extract 150 seconds from each of the remaining 80 songs based on the activity of all sources. This is done with the help of a tool provided by the creators of *musdb18*, which approximates the confidence of source activity as a function of time by applying a logistic function to the respective signal after half-wave rectifying, compressing, smoothing and down-sampling it (as described in [131]). The dedicated test data set holds 50 songs.

3.3.2 Pre-processing

In order to reduce computational cost, all data is downsampled from 44.1 kHz to 32 kHz and converted to mono, as we are only considering the single-channel case. Note that the downsampling causes the frequency, that corresponds to the bin halfway on the spectral range and therefore also to the crossover point of the multi-band model, to shift from approximately 11 kHz to 8 kHz. Since relevant information above 11 kHz is considered to be rather scarce anyway, this can be thought of as an enhancement of the baseline system. Training examples are constructed such that nearly all available training data is seen approximately once during one epoch. We generate training examples by mixing sources from random chunks of different songs and applying random gains in the interval [0.25, 1.25]. While this data augmentation scheme generates incoherent and unrealistic mixtures, it has been found to improve generalization [76]. The same technique, plus a random swap of the stereo channels which has to be neglected in this case, is also used in the MMDenseNet reference paper.

Spectrograms are obtained by stacked sequences of the short-time Fourier transform (STFT) using a frame and hop size of 2048 and 512 samples, respectively. The transformed STFT representation of the whole musdb18 audio corpus has been pre-computed and stored on disk to avoid unnecessary recalculations during training. It is not clear from the paper how many frames are fed to the MMDenseNet per pass. The follow-up paper [86] (MMDenseLSTM), however, states an effective context window size of 356 frames. As this work is supposed to consider resource-constrained environments and quasi real-time settings, we necessarily have to deal with rather small context sizes: each training example contains either 64 or 8 time frames, with the latter corresponding to experiments with reduced context size (176 ms). As the DC component of spectrograms carries no exploitable information, can simply be set to zero for the source magnitude estimates and smaller feature map sizes directly translate into reduced computational loads, it is discarded before being fed to the neural network.

3.3.3 Training

Although the original MMDenseNet trains with RMSProp, the follow-up MMDenseLSTM [86] utilizes the Adam optimization algorithm [132]. Since conformity with the latter framework is desirable as we want to maintain the option of inserting recurrent units at a later stage, and due to Adam being more stable in general, we stick to this scheme. Additionally, both adaptive optimization methods draw on similar ideas and the update rule of Adam essentially differs from that of RMSprop only by an additional exponentially decaying average of the past gradients (similar to *momentum*). In other words, while RMSprop does (uncentered) variance normalization of the gradient, Adam also accounts for the mean and therefore tends to accelerate the search in direction of the minima. We use the updated algorithm, termed *AdamW* [133], which, in contrast to the original version that accumulates L2 regularization terms in the calculation of the moving averages, properly applies weight decay only in the actual update step.

The training objective (for the baseline system at least), is to minimize the mean squared error (MSE) on the vocals linear magnitude spectrogram. The best model is chosen based on the minimal loss on the validation set. It is also possible to use the previously introduced BSSEval metrics (SDR, SIR, SAR) and use a possibly weighted combination to select the best performing model. However, calculating these ratios on the whole validation set is costly and doing so every epoch significantly extends the training process. Moreover, the loss probably represents a better overall and more interpretable measure than a weighted combination of the BSSEval metrics. Naturally these are used in the testing stage though, with the help of the provided *museval* python tool. The measures are calculated based on a time window of 1 s and allow a time-invariant distortion filter.

3.3.4 Hyperparameters

While the paper reports on parameters with regard to the network configuration, it omits hyperparameter settings such as values for the learning rate, batch size or the number of training epochs. Therefore a random search (rather than a grid search [134]) was conducted. As it is impractical to show the outcomes of all experiments in numbers and some of the parameters are also subject to additional theoretical and/or other practical considerations, the findings are presented in the following. A summary of the chosen hyperparameters is presented in Tab. 3.1.

Unlikely settings were ruled out after few epochs of training, before increasing the training time in order to fine-tune the parameters. Training in general took quite long, which made it hard to properly infer settings for e.g. early stopping. Rather than expensive parallel calculations on the GPU like e.g. backpropagation, the computational bottleneck was due to slow disk read data fetching operations, arguably caused by a slow filesystem. Because of the more or less heavy baseline system and the relatively long context needed, GPU memory lacked the space to transfer the entire dataset onto it, which would have rectified the problem. Therefore it was decided to simply choose a fixed number of 100 **epochs** before selecting the best model, since the validation loss has been found to converge sufficiently well over this period.

Traditionally one of the crucial parameters is the **learning rate**, even though modern adaptive optimization algorithms are designed to alleviate the need for elaborate tuning. The default value of Adam, 0.001, works well, however a minor increment up to 0.002 showed similar results yet faster convergence. Additionally, a schedule was employed to decay the learning rate by a factor of 0.1 when the validation error plateaus.

Being closely connected to the learning rate parameter, there is currently much debate about whether larger or smaller values for the **batch size** ought to be preferred [135], [136]. While the ultimate truth remains to be found, the existence of a generalization gap in favor towards smaller batch sizes is generally agreed upon [137], [138]. Adhering to smaller mini batches thus is certainly not detrimental to training. Additionally, Adam averages out noisy gradients in any case. Considering that the computational bottleneck corresponds to the loading of data and not the GPU, the batch size should not be too low either. A moderate size of 32 is used, which is also the maximum batch size of training data we could fit onto GPU memory.

It is less common to tune the value of the ϵ term added to the denominator of the Adam update step to improve numerical stability. However, large values of ϵ cause Adam to approach a behavior similar to standard stochastic gradient descent, and the overall stability of the optimization process increases as a trade-off for training speed (due to the smaller updates). In our case, it was observed that a slightly larger value of 1×10^{-6} yielded better performance than the default 1×10^{-8}.

The results of the hyperparameter search furthermore showed that the MMDenseNet is rather sensitive to regularization. Specifically, Dropout [139] almost completely prevents the model from learning except for small probabilities below 0.05 and has thus been rejected as regularization strategy. This behavior is expected though since the model makes extensive use of Batch Normalization: while the beneficial effect of BatchNorm is still not well understood [140], it is clear that Dropout causes an undesirable variance shift of neural units when turning from training to testing, since the fixed BatchNorm statistics are conditioned on slightly different activation distributions [141]. Furthermore, the dataset is rather small compared to others that are typically used in the field of image processing and therefore zeroing out activations, although intended to prevent overfitting, might as well have a detrimental effect. While strong **weight decay** is not appropriate either, the model is less sensitive to this regularization method and a moderate value of 1×10^{-5} showed satisfying results. Note that weight decay is not applied to bias terms (which are only used for the input and output layer) nor to the learnable parameters γ and β of BatchNorm layers. For insights concerning the interaction between BatchNormalization layers and weight decay, see e.g. [142], [143], [144]. In essence, the regularizing

effect of weight decay is reduced when used in combination with BatchNorm, but it is still beneficial as it helps to prevent the effective learning rate from decaying too fast.

Similarly, the paper does leave out the used initialization scheme. However, for residual networks with ReLU activations like DenseNets, it is common to use the so-called normally-distributed *Kaiming*-initialization scheme [145], which has been adopted for all experiments conducted throughout this work.

Table 3.1: Hyperparameters used for experiments throughout this work.

Parameter	Value
Epochs	100
Initial learning rate η	2×10^{-3}
Batch size	32
ϵ (numerical stability for Adam)	1×10^{-6}
Weight decay	1×10^{-5}

4 Performance

After introducing initial baseline results, this chapter considers the efficacy of several possible extensions in order to improve upon this baseline with regard to performance. Since these are not necessarily disjoint, they are presented in a successive manner with results presented intermittently. A concise summary of this chapter's findings is given in Section 4.7.

4.1 Baseline

Before building the eventual baseline system that should serve as the basis for this work, it is preferable to rule out additional potential shortcomings of the architecture first.

4.1.1 Input layer kernel size

Since the first layer is the most interpretable, it is interesting to tune its parameters as we can draw parallels between convolutional layers and kernel additive modelling techniques discussed in Section 2.5.3. The size of the initial convolution kernel is 3x3 in the reference paper. Because information in spectrograms is non-local and tends to be spread across both time and frequency, increasing the kernel size seems to be a natural choice.

However, it turns out that smaller filters work slightly better in practice. The experimental results shown in Tab. 4.1 indicate that models using smaller input kernels tend to produce better results while exhibiting comparable objective function costs. As features progress into the CNN, the receptive field increases and global concepts are captured in deeper layers of the model - the scale of the initial local features seems to have a minor impact in the process. Considering that the computational cost increases with employing larger convolutional kernels, it is clear that smaller kernels constitute a better overall choice for our purposes.

Table 4.1: Results for models using different kernel sizes in the input convolutional layer. Values correspond to the median value in dB over the musdb18 test set for the SDR, SIR and SAR and to the mean squared error for the objective function loss. Additionally, the number of model parameters and the number of multiply-and-accumulate operations (MACs) during inference for a single input example (batch size equal to 1) is given. K and G refer to the power-of-ten multipliers 10^3 and 10^9, respectively.

Kernel size	Loss	SDR	SIR	SAR	Parameters	MACs
19x19	0.360	5.04	9.68	6.04	289.569K	5.746G
7x7	0.363	5.07	9.59	6.10	259.617K	4.437G
3x3	0.364	5.14	9.61	6.29	255.777K	4.270G

4.1.2 Output non-linearity

The paper does not state whether the output comes directly from the last convolutional layer or if positive values are enforced, e.g. via a non-linear output function like a ReLU. Interestingly, the initial setup without any output function gives reasonable results, although bins with negative values are simply mirrored around zero to yield a valid magnitude spectrogram estimate for signal reconstruction. However, using a ReLU to enforce a positive output improves the results as expected, see Tab. 4.2. Thus, it has been decided that all networks that directly estimate spectrograms use a ReLU at the output throughout this work.

Table 4.2: Results for models using different output functions. Values correspond to the median value in dB over the musdb18 test set for the SDR, SIR and SAR and to the mean squared error for the objective function loss.

Output function	Loss	SDR	SIR	SAR
None	0.374	5.24	10.46	5.84
Absolute value	0.362	5.23	10.63	5.68
ReLU	0.360	5.27	11.02	5.61

4.1.3 Baseline performance

A detailed performance analysis of the baseline system is shown in Tab. 4.3. It is hardly possible to compare the result to the reference paper, which uses a slightly different data set (DSD100), a different sample rate, probably a different context size, considers the multi-channel case, and enhances its results using a post-processing multi-channel Wiener filter. However, it is noteworthy that the performance of this single-channel baseline implementation of the MMDenseNet achieves a median signal-to-distortion ratio which is approximately only 0.5 dB less than what is stated in the original publication, by directly estimating spectrogram magnitudes without additional post-processing steps and using a rather small context size. The corresponding training and validation error curves are depicted in Fig. 4.1, examples of magnitude spectrograms are shown in Fig. 4.2.

Table 4.3: Performance metrics yielded using source estimates from the baseline system with long context size (64 frames). Values correspond to the median, the median absolute deviation (MAD), the mean μ and the standard deviation σ in dB over the musdb18 test set. Additionally, the objective function loss is presented.

Metric	median	MAD	μ	σ
SDR	5.46	3.67	0.26	19.41
SIR	11.71	4.97	3.89	23.59
SAR	5.87	3.64	5.65	4.72
Loss			0.319	

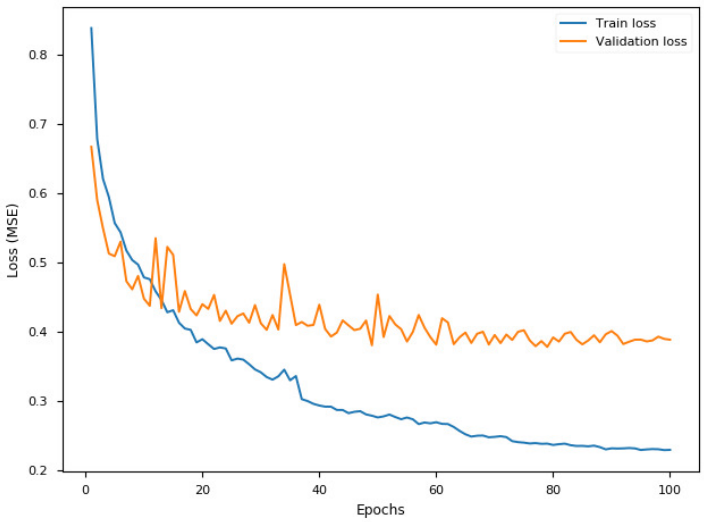
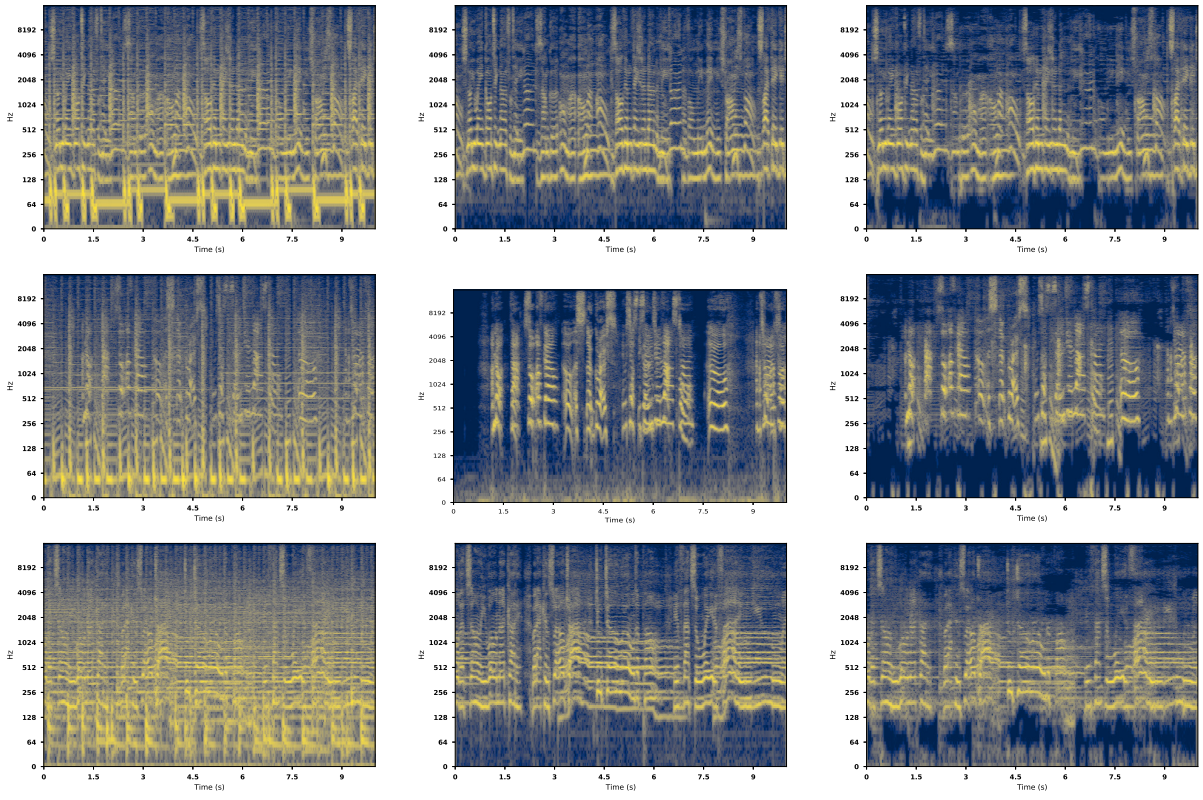


Figure 4.1: Training and validation objective function loss of the baseline system.



(a) Mixtures *(b) Targets* *(c) Estimates*

Figure 4.2: Spectrograms for the mixture (a), true vocals (b) and the baseline model's estimate (c) of 10 second extracts from 3 different tracks taken from the musdb18 test set. The frequency axis is scaled logarithmically.

4.1.4 Effect of logarithmic scaling

Human perception in general is proportional to the logarithm of a stimulus' actual intensity, as described by the Weber-Fechner law. The human ear is no exception: it is capable of processing stimuli over a large range of several orders of magnitudes (both in terms of amplitude as well as frequency), which are also perceived logarithmically. Hence the transformation of audio data onto the logarithmic scale is natural as well as convenient, since the large dynamic range can be represented on a more compact and reasonable numeric scale. Logarithmic magnitude spectrograms thus show a smoother distribution in contrast to their linear counterparts, which typically exhibit values very close to zero and large spikes in active frequency regions. In the case of data-driven source separation systems, training a model to produce reasonably distributed outputs seems desirable, as the corresponding representation might be easier to learn.

However, our experiments did not confirm that the baseline benefits from logarithmically scaled magnitude spectrograms. On the contrary, they prove to severely hurt the neural network's performance and occasionally prevent it from converging altogether. Many different setups have been considered to validate this claim and possible sources of errors have been carefully excluded. Tab. 4.4 presents the results that underpin these findings. Specifically, the training procedure is extremely sensitive if logarithmic target spectrograms are used (rather than logarithmic inputs and linear outputs). For this configuration, successful training was only possible using a very limited amount of training data, and including more data would prevent models producing logarithmically scaled outputs to converge. For this reason, Tab. 4.4 only shows experiments that overfit the model on a single training example. In addition, multiple training runs yielded wildly varying results for logarithmic source estimation, as can be seen in Tab. 4.4. This indicates that the initialization of the network parameters plays a crucial role.

Table 4.4: Results for overfitting the baseline model with different configurations for input and output scales. The train and test data set in this case consisted of the same single example: an excerpt from the musdb18 track "Music Delta - Beatles". The excerpt is 2 seconds long and during this time all sources are active. The "shift" parameter controls how much temporal variation is introduced when constructing a batch of training examples, each of length context size (which is 1.072 seconds in this case, or 64 spectrogram frames). A value of zero therefore implies that every epoch the exact same 64 frames are extracted from the 2 seconds multi-track recording. Higher values allow for more random variation from these static boundaries during data loading. No data augmentation is applied. To account for negative values due to the logarithm, no output non-linearities are employed, except when mask approximation is considered - in this case a final sigmoid layer is used. If the network is trained to approximate spectral masks while its output should be scaled logarithmically, an explicit conversion onto the linear scale is performed before applying the sigmoid non-linearity.

Input/Output scale	Epochs	Shift	Loss	SDR	SIR	SAR
log/lin	10	0	3.16	3.36	3.87	6.95
lin/log	10	0	1.25	-184.93	-22.12	-4.83
lin/lin	600	0	0.01	11.91	16.68	13.78
log/lin	600	0	0.01	11.94	16.79	13.81
log/log	600	0	0.09	7.42	16.77	10.54
lin/lin	500	1/4	0.25	10.69	15.66	11.95
log/log	500	1/4	3.87	-32.91	-9.17	-9.65
log/log	500	1/4	0.55	5.96	14.75	6.74
lin/lin (mask)	500	1/4	0.37	9.70	14.57	11.59
log/log (lin mask)	500	1/4	0.71	7.63	8.57	8.03

While the MMDenseNet's use of ReLUs may seem incapable of properly propagating features derived from logarithmically scaled magnitude spectrograms (which contain negative values in particular), this is not true since the ReLUs are paired with BatchNorm layers, that include learnable bias terms capable of shifting the distribution accordingly. Besides, possibly discarded information can easily be recovered due to the dense connectivity inside the network. Experiments using various activation functions (not included in Tab. 4.4) did not behave differently.

The poor training behavior can be explained by recognizing that the network is finitely accurate, and that noise in the estimate has a much larger impact in the logarithmic domain than the same noise in the linear domain. The larger error in the logarithmic domain thus may significantly impede optimization. This also explains the strong dependence on the initial conditions - if the network has a slightly disadvantageous configuration to begin with, the initially noisy estimate will have a stronger impact on the training if operated on the logarithmic scale.

Another way to look at this problem is to consider the contribution of the error between estimates and a target magnitude spectrograms for both domains. Assuming that an initial estimate roughly corresponds to normally distributed noise centered around zero - a reasonable assumption given the normally-distributed *Kaiming*-initialization scheme [145] - the error signal will be dominated by the large peaks of the *linear* target spectrogram's active time-frequency bins rather than its inactive bins with values close to zero. Therefore, in order to minimize the spectral distance, updates to the parameters of the estimator will rather force it to make better predictions for relevant regions than for unimportant ones. However, the *logarithmic* target spectrogram does not exhibit signal peaks as distinct as in the linear case. Magnitudes are represented on a more compact scale and the dynamic range between insignificant and relevant parts of the signal is reduced. It follows that the direction of the gradient will be less sharply oriented, since a larger number of (possibly irrelevant) bins contribute more to the error signal.

Note that while the results in Tab. 4.4 indicate that logarithmically scaled *inputs* work well, they have not been adopted as the default for the remainder of this work. Firstly, the network would have to learn an additional transformation without gaining any particular advantage. Secondly, the performance obtained from further experiments using logarithmic mixture magnitude spectrograms was not as satisfying as sticking entirely to the linear scale.

4.1.5 Mean-Variance normalization

It is common practice to rescale input features to have a mean of zero and unit-variance, which often leads to more stable training behavior and faster convergence. While the audio waveform is normalized on the interval $[-1, 1]$, the dynamic range of music signals and especially vocals is inherently very large. However, with large convolutional neural networks data normalization seems to be less of an issue since BatchNormalization layers assure well-behaved data distributions.

Indeed, experiments presented in Tab. 4.5 showed that mean-variance normalization along the frequency axis (per frequency bin) does neither benefit the performance nor the convergence speed. Apart from the effect of BatchNorm, another possible reason for this result could be that the original structure between different frequency bins is lost. The network should be able to resolve this effect though. Additionally, the hyperparameter search has been conducted for the configuration lacking the normalization, but we wouldn't expect the hyperparameters (e.g. learning rate and weight decay) to have a signficant impact on these findings. The best results are thus still attained with simple linear to linear spectrogram training.

Table 4.5: Results for the baseline system with and without mean-variance normalization of the input data for each frequency bin. Statistics are computed only on the training mixture data and kept fix during inference. Training targets are linear spectrograms in both cases, the model trained on normalized data however includes learnable parameters at the output to account for the scaling. BSSEval metrics correspond to median values in dB.

Mean-Var. Norm.	SDR	SIR	SAR	Loss
yes	5.35	11.12	5.91	0.343
no	5.46	11.71	5.87	0.319

4.2 Mask estimation

As discussed in Section 2.4, sources are separated by treating the audio mixture as wide-sense stationary and applying a time-varying filter or mask, typically only on the magnitude spectrogram. If source magnitude spectrograms are inferred directly from a DNN, the network itself can be considered as the filter, though the quality of the obtained results usually significantly benefits from a post-processing mask estimation step in this case. Indeed, [146] shows that mask-based DNNs are superior, at least with regard to speech separation, which may be attributed to the reduced search space during optimization and the presumably simpler structure of masks in comparison with spectrograms.

By contrast, a counterexample to this commonly accepted practice is presented as a preliminary experiment in [147], which shows that estimating magnitude spectrograms directly by a DNN more accurately describes the target source magnitude spectrogram in terms of MSE than its post-processed Wiener filtered version. While one has to acknowledge that the MSE is not a very good evaluation metric, especially with regard to perception, this result does nonetheless implicate that the best approach could be model dependent. Wiener filter masks are also bound to predict non-optimal estimates, since the assumption of additive power-spectrograms without destructive interferences does not hold in general, and the optimal magnitude exponent for the generalized Wiener filtering is unknown (see Section 2.4 or [19]). Additionally, as already addressed in Section 2.5.4, typically only one estimate per DNN is obtained. Wiener filtering requires an estimate of the mixture model though - that is, at least one estimate describing all accompanying sources - in order to produce non-trivial solutions, and employing an entire second DNN model only for better post-processing is computationally infeasible with regard to the efficiency objective of this thesis. While an estimate of the accompaniment could be obtained via spectral subtraction [18], this method usually only works well if target and accompaniment are approximately uncorrelated and have a mean of zero, and is thus more common for enhancing noisy speech.

But there are also other masks one could estimate. Naturally, the best mask for estimating the target source $\boldsymbol{S}$ in the TF-domain is the ideal complex (IC) mask $\boldsymbol{M}^{IC} = \frac{\boldsymbol{S}}{\boldsymbol{X}}$. The ideal amplitude (IA) mask $\boldsymbol{M}^{IA} = \frac{|\boldsymbol{S}|}{|\boldsymbol{X}|}$ only works well if the phase of the target source and the mixture is the same, which is generally not true. Interestingly, [149] takes into account that the majority of source separation algorithms only estimates the magnitude and applies the original mixture phase for reconstruction. If target and mixture phase are far apart from each other, estimating only the target magnitude without incorporating phase information can lead to large errors, as depicted in Fig. 4.3. Hence, the phase-sensitive (PS) mask $\boldsymbol{M}^{PS} = \frac{|\boldsymbol{S}|}{|\boldsymbol{X}|}\cos(\angle\boldsymbol{S} - \angle\boldsymbol{X})$ yields optimal results if the mixture phase is to be applied to the estimated source magnitude. Note that all of these three masks are unbounded, and thus do not exhibit the desired property of a constrained optimization space. Nonetheless, [149] shows that a truncated phase-sensitive mask (i.e. constrained to the interval $[0, 1]$) still produces favorable results, both as an oracle mask as well as a training target.

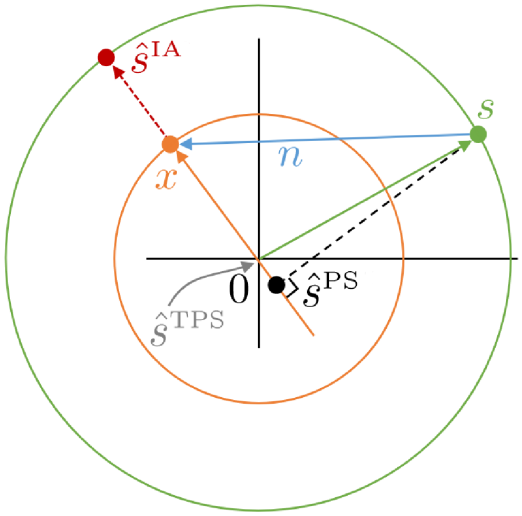


Figure 4.3: Illustration of the effect of using the ideal amplitude mask (IA), the phase-sensitive mask (PS) or the truncated phase-sensitive mask (TPS) for the estimation of $\hat{s}$, when the mixture phase is applied for reconstruction [148]. The mixture and the true target are denoted by x and s, respectively.

Furthermore, [149] states that minimzing the loss function between the target and the *masked estimate* (spectrogram approximation), is superior to directly optimizing for a particular mask (mask approximation). Thus, another intuitive and promising alternative, is to leverage the DNNs learning capabilities and let the network learn an appropriate mask itself, without explicitly specifying the mask. This idea has been already investigated in [79] and [80], and [92] also proved that this approach helps denoising auto-encoders (such as U-Nets) to learn non-trivial mapping functions. Only a minimal architectural change is required to accomplish this: a simple multiplication of the output of the DNN's last layer with the input mixture spectrogram. Additionally, it is possible to first prime the network by pre-training on a particular mask before inserting the multiplication, and afterwards discard the explicit conditioning on the mask and re-train on approximating the spectrogram with a learned mask. Both approaches have been explored in the following experiments.

4.2.1 Results

Separate spectrogram estimating models for vocals (Baseline, vocals) and accompaniment (Baseline, accomp.) serve as the basis for these experiments and have been trained according to the setup described in Section 3.3. The estimates obtained from these two models are used to calculate a traditional Wiener filter as a post-processing step as discussed above, for the singing-voice target (Wiener filter, vocals). Models that were trained on a mask approximation objective employ a sigmoid output non-linearity, instead of the ReLU function used for the models trained to approximate spectrograms, in order to match the output range to those of the mask values. The models are trained on pre-computed oracle masks, in particular the Wiener filter (WF-mask) or the truncated phase-sensitive mask (TPS-mask). Experiments that allow the DNN to learn a mask itself also feature the sigmoid output function, but the sigmoid output is multiplied with the input mixture in order to yield a masked spectrogram estimate (learned mask). Pre-training is achieved by training the model on the Wiener filter (WF) or truncated phase-sensitve (TPS) oracle masks for 40 epochs, before skip-connecting the input to the output and re-training on the target magnitude spectrogram for 60 epochs.

As expected, the post-processing Wiener filter yields significantly better results and is especially effective in reducing interfering noise. It also exhibits the lowest value for the signal-

Table 4.6: Results for experiments investigating the effect of different mask-based approaches. BSSEval measures correspond to the median value in dB over the test set.

Method	Output operation	Pre-Training	SDR	SIR	SAR	MSE
Baseline, vocals	ReLU	-	5.46	11.71	5.87	0.319
Baseline, accomp.	ReLU	-	12.95	17.44	15.27	0.281
Wiener filter, vocals	-	-	5.62	14.38	5.42	0.357
WF-mask	sigmoid	-	5.24	12.82	5.51	0.071
TPS-mask	sigmoid	-	5.22	11.11	5.96	0.076
learned mask	sigmoid (+ mult.)	-	5.48	10.87	6.17	0.323
learned mask	sigmoid (+ mult.)	WF	5.57	11.48	6.08	0.316
learned mask	sigmoid (+ mult.)	TPS	5.63	11.17	6.45	0.321

to-artifacts ratio though. Moreover, the claim made by [147], that direct estimation is more "accurate" than Wiener filtering, can be verified in terms of the MSE of the magnitude spectrograms. However, we see that the spectral Euclidian distance is not a good measure in this case and, as expected, the positive effect of the Wiener filter can be observed from the objective evaluation metrics.

The enhanced suppression of interfering signals through the Wiener filter objective can also be observed when training the network on the corresponding oracle mask. However, explicitly enforcing a mask approximation training scheme does not generally improve the separation performance, neither in the Wiener filter nor in the phase-sensitive case.

Given that the network configuration is essentially the same except for a single multiplication, it is not too surprising that the performance of the learned mask is rather similar to that of the spectrogram estimation approach. It is nevertheless surprising that the optimization procedure does not really benefit from the constrained search space, since the performance of both methods is rather similar. While the largest difference is the better SIR of the direct spectrogram filtering model, one could argue that the learned mask obtains estimates of superior signal quality, which is expressed in the SDR and the SAR.

However, the experiments clearly demonstrate that pre-training on oracle masks does indeed benefit the performance of mask estimation networks. Compared to the other models, both the WF- and TPS-pretrained models perform very well across all metrics. The latter even outperforms the Wiener filter, which is calculated using estimates from two distinct DNNs, in terms of signal quality (SDR + SAR). This result simultaneously shows the usefulness of the phase-sensitive mask as well as the advantage of not being too restrictive with regard to the learned mask. Furthermore, it demonstrates that making use of only one DNN model, instead of several ones in the post-processing Wiener filter case, is definitely competitive in terms of performance.

4.3 Multi-task training

The standard training objective usually only specifies to minimize the error between the target and the estimate. Considering the goal of source separation though, it would be desirable to enforce that the estimate should not only be close to the target source but also to be dissimilar to all other sources in order to facilitate source separation. However, an additional discriminative loss term that penalizes similarities between the accompaniment and the estimate will not be very beneficial, since there are many possible unmusical signals that satisfy this property (such as white noise or silence). For this reason, another strategy may be better suited.

With regard to the encoder-decoder structure of the network, it is clear that the model should learn some general representation of the audio material that lends itself to convenient separation. Thus, it seems reasonable to employ a second output section for the simultaneous estimation of the accompaniment, similar to [74], [124] and [150]. Doing so also allows for optimizing the additional discriminative cost as explained above, since the joint latent representation of the sources hints the gradients of the contrasting loss terms in the right direction (see e.g. also [74]). Estimating the accompaniment in a multi-task training fashion therefore should act as a regularization for the target source estimation, and also reduce interfering accompanying sources.

For the MMDenseNet architecture there are basically two choices for where to branch off the separate source outputs from a joint representation. One could either split right after the encoding path of the U-Net and use two distinct decoders for each of the sources, or share the decoder of the U-Net across both sources and only employ two separate output Dense-Blocks. In the latter case, which is depicted in Fig. 4.4, the regularization also effects the decoder and the joint representation is preserved almost over the entire network. Although this might seem to limit the DNN's capability to model a particular source, it intuitively makes sense that the acoustic model of one source should be able to at the same time represent an acoustic model of the corresponding accompaniment, only by changing - or essentially inverting - the weights of the output layer.

Of course, estimating two complementary sources opens up the possibility of using a Wiener Filter for post-processing, or to incorporate deterministic output layers into the network structure and jointly compute Wiener Filter-like masks for this purpose [74]. However, keeping the computational overhead in mind, it seems more appropriate to utilize the network output head of the accompaniment only as a regularizer during training, and discard it during inference. Nevertheless, Tab. 4.7 includes experiments that employ the explicit Wiener filter-style output structure (denoted by *joint mask comp.*). Similarly, it is also possible to train each output section using a mask-based approach, as discussed in Section 4.2.

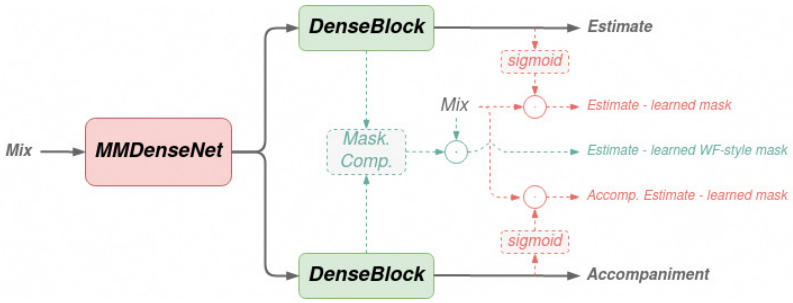


Figure 4.4: Illustration of the system configuration with separate output blocks for experiments concerning multi-task objectives. The blocks denoted DenseBlock correspond to the last densely connected block of the MMDenseNet. Dashed lines are optional and correspond either to mask-based experiments as done in Section 4.2 (red), or to explicit Wiener filter-like mask computation (blue).

4.3.1 Results

Comparing the baseline model with the model that uses two distinct decoding paths for estimating the two sources in Tab. 4.7, we can see that their performance is almost equal. Quite surprisingly though, the SIR is slightly worse in the case of the multi-task objective, but the differences are rather insignificant. Despite the significant increase of the model size in the case of two separate decoders, not even a combination of their outputs using the Wiener filter-like joint mask computation clearly improves the result, especially when compared to the single-task learned mask methods. For this reason, further experiments only focused on multi-task configurations that segregate right before the output Dense-Block. One interesting observation for both

Table 4.7: Results for different multi-task training configurations on singing-voice separation. The evaluation metrics for the accompaniment are omitted, since they are not of concern for this analysis. The first four systems correspond to the baseline system as well as the three systems using learned masks from the previous section Section 4.2.1. "Split" denotes which point the network is branched off into two separate output heads, after the U-Net encoding path or before the last dense block (as shown in Fig. 4.4). SA and MA denote spectrogram approximation and mask approximation, respectively.

Split	Target	Output op.	Pre-Training	SDR	SIR	SAR	MSE
-	SA	ReLU	-	5.46	11.71	5.87	0.319
-	SA	sigmoid (+ mult.)	-	5.48	10.87	6.17	0.323
-	SA	sigmoid (+ mult.)	WF	5.57	11.48	6.08	0.316
-	SA	sigmoid (+ mult.)	TPS	5.63	11.17	6.45	0.321
Decoder	SA	ReLU	-	5.49	11.28	6.01	0.296
Decoder	SA	ReLU + joint mask comp.	-	5.55	11.77	6.14	0.312
Out-Block	SA	ReLU	-	5.45	10.74	6.18	0.303
Out-Block	SA	ReLU + joint mask comp.	-	5.47	11.74	5.90	0.329
Out-Block	MA, WF	sigmoid	-	5.32	12.28	5.68	0.083
Out-Block	MA, TPS	sigmoid	-	5.02	12.55	5.49	0.085
Out-Block	SA	sigmoid (+ mult.)	WF	5.54	11.02	6.22	0.287
Out-Block	SA	sigmoid (+ mult.)	TPS	5.54	11.55	6.13	0.293
Out-Block	SA	ReLU + joint mask comp.	SA	5.63	12.37	5.92	0.316

multi-task configurations can be made though: comparing only the non-mask-based approaches (only ReLU output), the SAR is increased in both multi-task cases. This is expected as the shared general representation across the network should be more robust to introducing artifacts compared to the single-task models.

However, the multi-task training objective does not seem to be very beneficial in general, with maximum improvements over the single-task systems in the range of 0.1 dB. The baseline model even performs slightly better than the model with a shared decoder, and also shows a higher SIR. Additionally, analogous to the single-task case discussed in Section 4.2, direct mask approximation methods underperform, while pre-training schemes with learned masks work quite well. For the multi-task configuration it is also possible to pre-train on spectrogram approximation, and subsequently re-train using joint mask computation with a Wiener Filter-structured output. While this method yields decent performance, specifically in terms of SIR, its usage over the single-task model pre-trained on phase-sensitive masks is hardly justified, given the better SAR of the latter model and the increased computational cost for the estimation of the accompaniment source in the multi-task Wiener Filter-style model.

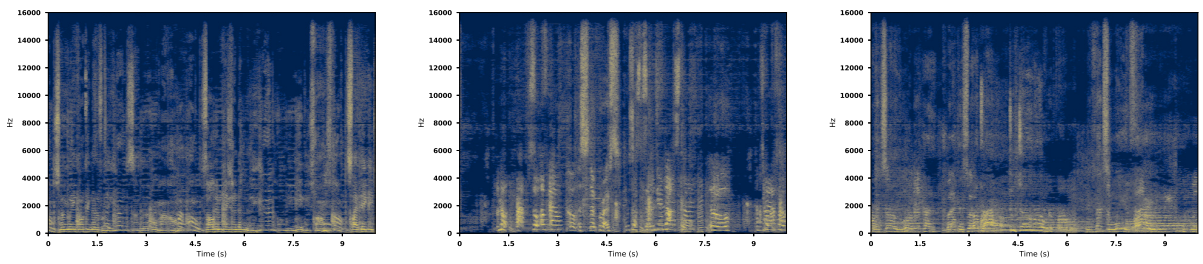
4.4 Redesigning the multi-band structure

A key contribution of the MMDenseNet paper [84] is the investigation towards utilizing multi-band structures instead of equally processing the full-resolution input all at once. It tries to compensate for the fact that different frequency bands exhibit different patterns in the spectrogram. The low-frequency regions of a singing-voice, for instance, will be dominated by components of the fundamental frequency, the first few harmonics as well as formants caused by resonances in the vocal tract, while less overall energy will be present in higher frequencies. Similarly, bass instruments contain virtually no information at all above a certain frequency, except for possible noise-like transients (e.g. noises from plucked strings). In addition, the spacing between harmonics increases along the frequency range, independently of the considered source. Since

convolutional kernels are shared across the entire input but are applied locally on different regions, frequency-dependent patterns may impede the performance of full-band CNNs. For this reason, the MMDenseNet comprises 3 distinct MDenseNets - one conditioned on the lower frequency band, one learning the spectral patterns of the higher frequency region, and another one for the full band in order to roughly model the global structure as well. Nevertheless, patterns in the TF-domain are still closely related to each other. Excellent examples are again harmonic components, which experience a so-called common fate, since they have strong affinities across frequency bands as well as with regard to their temporal development. Thus, patterns do show a certain amount of translation, and the strict segregation of the MMDenseNet structure does not seem optimal. Furthermore, the comparison between the single-band and the multi-band models conducted in [84] is not completely fair, since the number of parameters between them - and thus their respective potential expressiveness - is not well-balanced. The original MDenseNet uses only about 121×10^3 parameters, which is roughly 45 % of the amount of parameters that its multi-band counterpart has available (269×10^3).

Several potential flaws of the architecture can be identified. First, no information is shared between the distinct band-limited components of the MMDenseNet. While it makes sense to train kernels only on locally constrained frequency regions, ignoring information about the global structure when doing so appears to be a limiting factor. Second, the band-limited outputs are combined through a single Dense-Block before the final output layer. Although the three separate MDenseNets are trained to jointly produce reasonable and consistent outputs, reconstructing a full-band feature map as input to this "merging" Dense-Block is done by simply concatenating the single-band activations along the frequency dimension, which seems like a representational bottleneck. Third, the band limits appear to be chosen according to implementational convenience rather than prior knowledge about the musical sources, since a "cutoff" frequency of 8 kHz or even 11 kHz (for sampling rates of 32 kHz and 44.1 kHz, respectively) may split the spectrogram into two equally sized parts, but does not account for the fact that the higher frequency band will carry virtually no relevant information about the musical source. Splitting the spectrogram into two bands would be much more appropriate in the range of $1 - 4$ kHz for most music signals and would also correspond better to the human auditory perception of frequencies. In addition, only hard boundaries without any frequency overlap are considered. In fact, re-scaling the baseline spectrogram estimates of Fig. 4.1 onto a linear frequency axis reveals the effects of the MMDenseNet's rigid band-limited processing, as shown in Fig. 4.5, which exhibits slight boundary discontinuities at the cutoff frequency.

Although the multi-band idea seems sensible in general, its realization could be improved. Two approaches for re-designing the multi-band structure of the architecture have been investigated and are outlined in the following.



(a) *(b)* *(c)*

Figure 4.5: The 3 baseline estimates from Fig. 4.1 rescaled onto a linear frequency axis. By taking a closer look, thin horizontal lines at 8 kHz (cutoff frequency) can be detected.

4.4.1 Multi-band Channels

One possible approach is to arrange the band-limited spectrogram components in the channel direction and feed this "multi-channel" input to a MDenseNet (MBC). This way, information about all frequency bands can be captured and exploited by a single network, which is able to model the bands individually at the same time. This configuration is depicted in Fig. 4.6. However, this configuration suffers from one serious flaw: convolutional kernels operate on features that are *local across the channel direction*. Splitting the input spectrogram into several parts and putting them on top of each other destroys the spatio-spectral relationships between these individual components. Information about several frequency bands collapses into a single bin of a feature map and, in turn, cannot be be straightforwardly restored. An additional 2D-convolutional layer with a kernel size of 3×3 is employed for this reason that is supposed to be able to segregate feature maps according to the frequency bands.

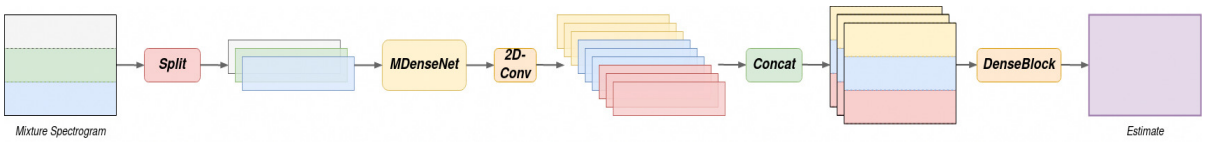


Figure 4.6: Illustration of the MDenseNet adapted for incorporating multi-band information along the channel direction (MBC). The last DenseBlock inside the MDenseNet is removed and instead corresponds to the last DenseBlock depicted here.

4.4.2 Multi-band Dense-Blocks

A different and apparently better approach would be to restrict kernels to certain bands but interchange information between them through shared activations. In practice, sharing feature maps is thus only possible between convolutional layers assigned to one particular band and that of the full band, since other layers will be assigned to other frequency regions (except for potential overlaps). Sticking to the general network MDenseNet architecture, this can be accomplished by turning each layer inside a densely connected block into a parallel multi-band processor. The layers still consist of functions consisting of the composite operation of *BatchNorm+ReLU+2DConv*, but now there are several of these composite functions per layer, and each one operates on a certain frequency region. The idea is depicted in Fig. 4.7, for the case of 2 bands. First, the input is split, or filtered, into several frequency bands, including the full-band. Then, the composite functions are applied to the band-limited input components. These can

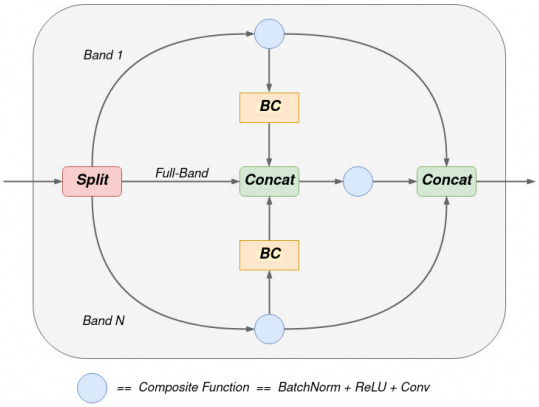


Figure 4.7: Illustration of a single composite function inside a densely connected block adapted for multi-band processing (MBDB). BC stands for bottleneck compression.

then be concatenated along the frequency axis to yield full-band sized feature maps, which can in turn be concatenated along the channel axis with the full-band activations coming from the input. This way, the following full-band convolutional kernels can incorporate information from the band-limited ones. In an attempt to let the band-limited kernels learn separate representations in an unobstructed way, their outputs first pass a 2D-convolutional layer of kernel size 1×1 (point-wise convolution) in order to transform these different activations onto a common representation before they are concatenated and fed into the full-band function. These 1×1-convolutions only act in the channel direction and can thus also be used to "bottleneck-compress" (BC) the activations in order to reduce the computational load, by effectively compressing the number of feature maps. This seems reasonable given that the following full-band layer will be rather computationally expensive. The output of this multi-band unit is then constructed by concatenating the full-band function's output with that of the band-limited ones. This output concatenation operation ensures not only that the full-band function of the next multi-band unit has access to all (both band-limited and full-band) activations produced in the previous multi-band unit, but also that the band-limited functions in the next unit have access to both the band-limited and the full-band activations of the previous unit. This guarantees proper exchange of information across global and local frequency regions and enables fine-grained dense connectivity between the corresponding composite functions. Similar to the MMDenseNet, this model makes it possible to model different frequency regions with different detail and compute power, by changing the number of kernels to learn per band.

This multi-band framework is rather flexible. For instance, one may decide whether only full-band features are passed to the next (down- or up-sampled) densely connected block, or if the band-limited feature maps propagate through the whole network. The former approach allows to change the cutoff frequencies and the number of bands that is considered at different scales of the network, which may also be useful in terms of computational efficiency. Low-resolution activations at lower scales in the network do not need to be processed with a lot of different band-limited kernels, for example.

4.4.3 Results

In order to conduct a fair comparison between the models, it has been taken care of that all models in total have approximately the same number of parameters as the baseline system. Tab. 4.8 sums up all different tested model configurations with regard to systems that use multi-band dense blocks. "BC kernels" denotes the number of bottleneck-compression kernels (of point-wise convolutional layers), which are used to transform different feature maps onto similar representations before concatenation, and therefore also describe how many feature maps are shared between full-band and band-limited kernels. Feature maps can either be shared at the input of a DenseBlock, at its output, or at all intermediate multi-band layers. Additionally, a dense block's output may be forced to a full-band representation with a dedicated full-band layer which all feature maps have to pass, as described in the previous paragraph. The number of overlapping bins is simply calculated by a multiplication of the cut-off bin's index with an overlapping factor. The weight given to overlapping bins of a band decreases linearly. Since these models are a bit more memory-demanding during training, the batchsize has been scaled down from 32 to 8. The parameters for the model with multi-band channels (MBC) is given in Tab. 4.9. The results are shown in Tab. 4.10.

Differences in performance between the various multi-band models are within a very narrow range of approximately $\pm 0.2\,\mathrm{dB}$, only the interference reduction seems to fluctuate a little more. While previous experiments did not exhibit highly pronounced performance peaks either, in contrast to the reference paper [84] this result indicates that multi-band processing is not overly relevant for the separation of musical signals.

Table 4.8: Different configurations for experiments with multi-band dense blocks (MBDB). The model is built symmetrically, which means that the given parameters apply both to the encoding as well as the decoding path of the network.

Model	*s*	*bands*	*k*	# input kernels	*L*	overlap	*shared feat.-maps, # BC kernels*			*Forced full-band output, # kernels*	*output block*	
							input	internal	output		k	L
MBDB-1	1	$0-8\,\text{kHz}$ $8-16\,\text{kHz}$ full	10 4 6	32 32 32	3	0.1	-	4	8	10	6	2
	$\frac{1}{2}$	$0-8\,\text{kHz}$ $8-16\,\text{kHz}$ full	10 4 6	-	4	0.1	-	4	8	10		
	$\frac{1}{4}$	$0-8\,\text{kHz}$ $8-16\,\text{kHz}$ full	10 4 6	-	4	0.1	-	4	8	10		
	$\frac{1}{8}$	$0-8\,\text{kHz}$ $8-16\,\text{kHz}$ full	10 4 6	-	4	0.1	-	4	8	10		
MBDB-2	1	$0-250\,\text{Hz}$ $250-1000\,\text{Hz}$ $1-4\,\text{kHz}$ $4-16\,\text{kHz}$ full	5 9 9 5 6	14 18 18 12 20	3	0.1	-	4	8	10	6	2
	$\frac{1}{2}$	$0-1\,\text{kHz}$ $1-4\,\text{kHz}$ $4-16\,\text{kHz}$ full	7 7 4 6	-	4	0.1	-	4	8	10		
	$\frac{1}{4}$	$0-8\,\text{kHz}$ $8-16\,\text{kHz}$ full	7 4 6	-	4	0.1	-	4	8	10		
	$\frac{1}{8}$	full	6	-	4	-	-	-	-	-		
MBDB-3	1	$0-8\,\text{kHz}$ $8-16\,\text{kHz}$ full	10 4 6	32 32 32	4	-	-	2	-	-	4	2
	$\frac{1}{2}$	$0-8\,\text{kHz}$ $8-16\,\text{kHz}$ full	10 4 6	-	4	-	-	2	-	-		
	$\frac{1}{4}$	$0-8\,\text{kHz}$ $8-16\,\text{kHz}$ full	10 4 6	-	4	-	-	2	-	-		
	$\frac{1}{8}$	$0-8\,\text{kHz}$ $8-16\,\text{kHz}$ full	10 4 6	-	3	-	-	2	-	-		
MBDB-4	1	$0-1\,\text{kHz}$ $1-4\,\text{kHz}$ $4-16\,\text{kHz}$ full	10 10 5 6	20 20 18 20	3	-	-	4	-	-	6	2
	$\frac{1}{2}$	$0-1\,\text{kHz}$ $1-4\,\text{kHz}$ $4-16\,\text{kHz}$ full	9 10 5 6	-	4	-	-	4	-	-		
	$\frac{1}{4}$	$0-1\,\text{kHz}$ $1-4\,\text{kHz}$ $4-16\,\text{kHz}$ full	9 9 5 6	-	4	-	-	4	-	-		
	$\frac{1}{8}$	$0-1\,\text{kHz}$ $1-4\,\text{kHz}$ $4-16\,\text{kHz}$ full	9 9 5 6	-	4	-	-	4	-	-		

Table 4.8: (continued)

Model	*s*	*bands*	*k*	# input kernels	*L*	overlap	*shared feat.-maps, # BC kernels*			*Forced full-band output, # kernels*	*output block*	
							input	internal	output		k	L
MBDB-5	1	0 – 8 kHz	14	16	4	-	3	-	5	-	4	2
		8 – 16 kHz	13	16	3							
		full	8	16	3							
	$\frac{1}{2}$	0 – 8 kHz	14	-	4	-	3	-	5	-		
		8 – 16 kHz	13		3							
		full	8		3							
	$\frac{1}{4}$	0 – 8 kHz	14	-	4	-	3	-	5	-		
		8 – 16 kHz	12		3							
		full	8		3							
	$\frac{1}{8}$	0 – 8 kHz	14	-	4	-	3	-	5	-		
		8 – 16 kHz	12		3							
		full	8		3							
MBDB-6	1	0 – 1 kHz	14	16	4	0.1	3	-	5	-	4	2
		1 – 4 kHz	14	16	3							
		4 – 16 kHz	8	12	2							
		full	6	16	3							
	$\frac{1}{2}$	0 – 1 kHz	9	-	4	0.1	3	-	5	-		
		1 – 4 kHz	10		3							
		4 – 16 kHz	5		2							
		full	6		3							
	$\frac{1}{4}$	0 – 1 kHz	9	-	4	0.1	3	-	5	-		
		1 – 4 kHz	9		3							
		4 – 16 kHz	5		2							
		full	6		3							
	$\frac{1}{8}$	0 – 1 kHz	9	-	4	0.1	3	-	5	-		
		1 – 4 kHz	9		3							
		4 – 16 kHz	5		2							
		full	6		3							
MBDB-7	1	0 – 1 kHz	10	16	4	0.1	5	-	-	-	4	2
		1 – 4 kHz	12	16	4							
		4 – 16 kHz	6	12	2							
		full	7	16	3							
	$\frac{1}{2}$	0 – 1 kHz	10	-	4	0.1	5	-	-	-		
		1 – 4 kHz	12		4							
		4 – 16 kHz	6		2							
		full	7		3							
	$\frac{1}{4}$	0 – 1 kHz	10	-	4	0.1	5	-	-	-		
		1 – 4 kHz	12		4							
		4 – 16 kHz	6		2							
		full	7		3							
	$\frac{1}{8}$	0 – 1 kHz	10	-	4	0.1	5	-	-	-		
		1 – 4 kHz	12		4							
		4 – 16 kHz	6		2							
		full	7		3							

Table 4.9: Configuration of the MDenseNet exploiting multi-band patterns in the channel direction (MBC). The values of k and L are used on each scale of the network. All other parameters follow the configuration of the baseline model. Note that this model does not use a full-band channel.

Cutoff	# input kernels	k	L	2D-conv assigning channels to bands, # kernels per band
8 kHz	96	16	4	16

Table 4.10: Results for experiments investigating alternative multi-band designs.

Model	SDR	SIR	SAR
Baseline	5.46	**11.71**	5.87
MBDB-1	5.44	11.23	5.98
MBDB-2	5.34	11.24	5.93
MBDB-3	5.40	10.74	**6.12**
MBDB-4	5.32	11.49	5.82
MBDB-5	5.34	10.83	6.04
MBDB-6	5.26	10.88	5.86
MBDB-7	5.34	11.27	5.83
MBC	**5.53**	11.59	5.90

None of the MBDB configurations performs better than the baseline system (except sometimes in SAR), although different aspects such as overlapping bands, intermediate sharing of information, and consistent full-band output representations have been considered, for instance in MBDB-1. MBDB-2, a configuration which makes heavy use of multi-band information with adaptive band-limits across different scales and varying emphasis on different bands, even performs a little worse than MBDB-1 which only uses fixed band-limits in combination with the remarkably high cutoff at 8 kHz. Both of these models enforce a full-band output representation with a separate composite function at the end of each dense block. From looking at the performance of these two models alone, it might seem as if their capabilities in modeling the distinct frequency bands is too limited. In particular, the variation of band limits (cutoff frequencies) across different scales in the case of the MBDB-2 might appear to be unfavorable, and learning many different band-limited kernels seems to hamper performance. However, using fixed bands and allowing the band-limited feature maps to propagate through the whole network, as done in MBDB-3 and MBDB-4, does not appear to be beneficial either.

In an attempt to tackle this circumstance, experiments using MBDB-5,6 and 7 investigate another approach to share features between band-limited kernels. Possibly the increased feature flow inside densely connected blocks prevent the individual layers from exclusively modeling a certain frequency pattern in detail. These configurations only share features at the input and/or at the output of a dense block. This also allows to use a different number of layers L for each frequency band in each scale. Nonetheless, these models even tend to perform a little bit worse.

Given the small differences in performance and the inconsistent results of the MBDB configurations, interpreting the results according to multi-band processing capabilities does not appear to be very conclusive. The results may be explained much better by considering how the different models distribute computational power to certain parts in the network. As stated above, the number of parameters of all models has been tuned to be approximately equal. However, while this was done in order to enable a fair comparison between the models, the necessary changes of the model configurations in turn affect the structure and the relative expressiveness of the models. For instance, the configurations MBDB-2,4,5,6 and 7 use fewer input kernels per band, and exhibit a lower average signal-to-distortion ratio. The number of input kernels per band therefore may be significant with regard to the performance of the model. Similarly, MBDB-2 and 4 only use 3 layers for the full-resolution dense blocks, while for instance MBDB-3 employs 4 layers. Different growth rates per block and per scale also seem to be very important for the network's performance. These structural differences make it very hard to properly compare the different models, especially because the MBDB configurations have many different tunable parameters. A thorough comparison between the models would thus be extremely time- and resource-consuming, and has not been conducted for these reasons. Another, albeit less con-

vincing explanation of the results is that all other hyperparameters are tuned for the original MMDenseNet and have been directly applied to the derived models, without explicit tuning.

Nevertheless, the results do indeed indicate that multi-band processing may not be very important with regard to music sources. This conclusion is also backed by experiments with the MBC model which, quite surprisingly, obtains the best overall performance measured via SDR, although the model destroys the spatio-spectral relationship between the TF-bins. Either the MBC model is able to separate between different frequency bands along the channel direction by learning to constrain the weights of kernels to certain input channels, or the limited weight sharing scheme according to different frequency bands is not particularly beneficial. In order to resolve this matter, the distribution of the kernel weights can be investigated. The additional 2D-convolution - which is used before grouping the channels into different bands via a concatenation of the feature maps along the frequency axis - may be particularly informative. If the kernels of this layer can be grouped according to the input channels, it is clear that the network learns to separate different frequency bands into different channels. Because the way the output feature maps of this layer are grouped and concatenated is known, it is also possible to assign a layer to each kernel. Fig. 4.8 depicts a principal component analysis (PCA) performed over the ℓ_2-norms of all 3×3 weight matrices connected to particular input channels for each output kernel of the layer. It turns out that these kernels can not be straightforwardly clustered, indicating that the MBC configuration does not learn band-limited kernels at all, while still obtaining a slightly better performance. Of course, the increased SDR could also stem from the relatively high number of input and output kernels on the full resolution scale, as discussed above. Despite the slightly better performance, this model has not been investigated in further experiments, not only because of the weak theoretical justification but also due to its increased memory consumption and inference times.

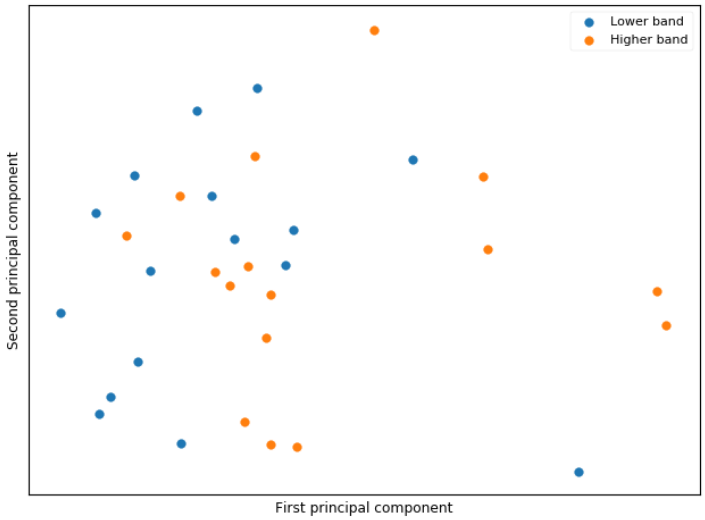


Figure 4.8: PCA performed over norms of weights of the kernels from the additional convolutional layer used to reconstruct the different frequency bands from the channels, corresponding to different input channels.

4.5 Exploiting and estimating phase information

Considering that phase information seems to be particularly relevant to perception [24], [25], simply applying the mixture phase for reconstruction does seem like a very limiting factor. Moreover, this entails limitations with regard to the accurate estimation of the source magnitude,

as already explained in the discussion of the phase-sensitive mask in Section 4.2 and illustrated in Fig. 4.3.

As demonstrated by the very renowned *Griffin-Lim* algorithm [151], a phase estimate can be reconstructed directly from a magnitude spectrogram by enforcing consistency and exploiting the redundancy of the STFT. While the original algorithm is an iterative one, the approach has been extended to be efficient and applicable in on-line processing environments [152], [153]. But there are also other non-iterative and real-time capable methods that obtain phase estimates from STFT magnitudes [154]. Such phase reconstruction techniques may be applied in a post-processing step, but significant and objectively measurable improvements in separation quality will in general not be expected from these methods. Rather than promoting a slightly more consistent and better explainable result from the unaltered magnitude estimate, it is desirable to directly include information about the clean phases in a data-driven way to get a better overall source estimate in the first place. An estimate of the target phase also allows to directly minimize e.g. the ℓ_1-norm in the waveform domain or a performance-based cost function such as the scale-invariant SDR. In this context, it also seems reasonable not to discard but to utilize the mixture phase information in some way to get a better separation.

DNNs operating in the time-domain have become increasingly popular and are able to achieve good separation quality, which may at least be in part attributed to their implicit ability of capturing phase information. But, as already stated in Section 3.1, time-domain models learn transformations that bear strong resemblance to the Fourier Transform, but can in general not be calculated as efficiently.

Utilizing complex values seems like a very natural way in order to both exploit and estimate phase information. [155] has taken a first step in this direction by estimating the real and the imaginary components as separate training targets and uses these to construct a complex ratio mask. While this approach uses a real-valued neural network, fully complex-valued neural networks with complex-valued activations, weights and gradients, also have been investigated for source separation and related tasks [156], [157], [158]. However, the performance improvements caused by complex-valued DNNs are not substantial, and the increased amount of operations as well as the additional effort regarding implementation do not seem to justify their use. A similar argument against complex-valued DNNs is made by [159].

The cyclic nature of the phase makes it hard to directly use it as an input feature or training target. An unwrapped phase is not very useful, since the increasing values over time prevent the construction of a meaningful and practical representation for pattern recognition. On the other hand, the discontinuities of a wrapped phase, with values in the interval $[-\pi, \pi)$, destroy its structure and let phase information appear to be rather random. A more useful, alternative representation of the phase (denoted by ϕ) may be obtained by means of its gradient with respect to time and frequency. The instantaneous frequency $\frac{\partial}{\partial t}\phi$ and the group delay $\frac{\partial}{\partial f}\phi$ can be successfully utilized as input features for music source separation, as shown in [160]. The gradient can be readily computed from the properly unwrapped phase spectrogram via finite differences along the time- and frequency-axis. Naturally, the phase is sensitive to shifts of the signal. With respect to the discrete Fourier transform of length N, this property is expressed by the shift theorem

$$\mathcal{DFT}\{x(n - n_0)\} = \mathrm{e}^{i\frac{2\pi}{N}kn_0}X(k) \tag{4.1}$$

which states that a shift n_0 in the time-domain corresponds to a linear phase term in the frequency domain for each frequency indexed by k. In order to yield compact and comparable feature distributions along the different frequency axis, the instantaneous frequencies (phase derivatives with respect to time) have to be corrected according to the phase shift caused by the hopsize of the STFT (i.e. $n_0 = hopsize$). Additionally, both [154] and [160] point out that

for a continuous-time STFT obtained with a real-valued Gaussian window of time-frequency ratio support γ, the log-magnitude $\log(|X(\omega,t)|)$ and the phase $\phi(\omega,t)$ are related through their gradients via the following relationships:

$$\frac{\partial}{\partial\omega}\phi(\omega,t) = -\gamma\frac{\partial}{\partial t}\log(|X(\omega,t)|) \tag{4.2}$$

$$\frac{\partial}{\partial t}\phi(\omega,t) = \frac{1}{\gamma}\frac{\partial}{\partial\omega}\log(|X(\omega,t)|) + 2\pi\omega \tag{4.3}$$

This indicates that features derived from phase information may help to estimate the magnitude, and vice versa, magnitude estimation may be useful to estimate values of the target phase. Thus it seems reasonable to share features between computational units, when each of these is assigned to only one of the two tasks.

While this representation could in general also be used as a training target of a phase estimation network, it seems to be unnecessarily indirect. Instead, [147] and [148] explored the use of a classification approach with discretized phase values as classes for each TF-bin. The output of a (real-valued) network then corresponds to probability vectors for each TF-bin, indicating which class is likely to give a good estimate of the target phase. The classes or, more specifically, the pre-defined discrete values on the unit circle (also called *codebook*) may simply be constructed from uniform sampling on the unit circle, but [148] also explores the use of other, in particular learned codebooks. Additionally, instead of simply picking the single most likely phase value ("arg-max"), [148] proposes to obtain continuous phase estimates through a weighted sum over the codebook, where the weights correspond to the probabilities of the respective discrete value: if P denotes the size of a codebook $\mathcal{P}_P = \{\phi_0 \dots \phi_{P-1}\}$, and $\boldsymbol{z}^{t,f} = \{w_0 \dots w_{P-1} \in \mathbb{R}^P \mid \sum_{i=0}^{P-1} w_i = 1;\ 0 \leq w_i \leq 1\ \forall i\}$ denotes the soft-max probability output vector of the network corresponding to the time-frequency bin indexed by t and f, then this summation can be given as

$$\hat{\phi}_{t,f} = \angle \sum_{i=0}^{P-1} \boldsymbol{z}_i^{t,f}\, \mathrm{e}^{i\phi_i} \tag{4.4}$$

Since this "interpolation" is done in the complex plane, this method, apart from gaining a continuous representation of the estimate, elegantly allows to interpolate along the unit circle, and hence introduces a grouping of the probability vector's elements according to the proximity of their codebook entries on the unit circle without any discontinuity. It is also straightforwardly applicable to mask estimation and provides a generalization of classical sigmoid activations.

4.5.1 Results

The concept of the following experiments is based on these considerations and is illustrated in Fig. 4.9. Instead of jointly modeling magnitude and phase in an intertwined manner (as is the case for complex-valued networks), an additional separate phase estimation DNN is employed. This also enables to balance computational power according to the relative relevance of the magnitude and phase estimators - denoted "MagNet" and "PhaseNet" - e.g. by reducing the size of the phase estimating model. The MDenseNet architecture is used as the basis for the PhaseNet, operating on only 3 different scales, each with $L = 3$ and $k = 4$ and only 16 kernels in the input layer. Its input is a concatenation of the instantaneous frequency $\Delta_{t,\phi}$ and the group delay $\Delta_{f,\phi}$, calculated using centered finite differences from the unwrapped phase spectrogram, and pre-processed as outlined in [160] (except for a twice as large phase shift compensation of $\frac{4\pi}{N}kn_0$ to account for the used *centered* differences). Bearing the relations of Eq. (4.2) and Eq. (4.3) in mind, the two networks exchange information before their respective output layers by a concatenation of feature maps in order to aid the estimation process. Specifically, the last

densely connected output block of the PhaseNet gets the estimated target magnitude scaled by the mixture magnitude $\log\left(\frac{|\hat{\boldsymbol{S}}|}{|\boldsymbol{X}|}\right)$ as input, a procedure that draws inspiration from [147] and aims at providing the PhaseNet with information about which of the mixture phase input values might be dominated by the target source and which by interfering sources. The task is formulated as a classification problem and a codebook of size $P = 8$, with discretized values uniformly spaced on the unit circle in the interval $[-\pi, \pi)$, is used with the interpolation method as described above.

Because an estimate of the true phase is now available, it is advisable to use the ideal amplitude mask $\boldsymbol{M}^{IA} = \frac{|\boldsymbol{S}|}{|\boldsymbol{X}|}$ as training target for the magnitude network. Note that the values for this mask are not bounded to the interval $[0, 1]$, and a simple sigmoid output non-linearity is not appropriate in this case. While a "stretched" sigmoid $f(x) = \frac{R_{max}}{1+\mathrm{e}^{-x}}$ or a "truncated" ReLU $f(x) = min(max(0, x), R_{max})$ may be used, where R_{max} denotes the maximum estimated mask value, it is intuitive to use the classification formulation in combination with interpolation over a codebook $\mathcal{M}$ also for the magnitude part of the system, since it generalizes the other approaches. Similar to [148], the codebook $\mathcal{M} = \{0, 1, R_{max} = 2\}$ is used for the following experiments.

Another advantage of having an estimate of the phase is the broadened range of possible loss functions. Since the inverse STFT (iSTFT) can be formulated using real-valued operations using a basis matrix that incorporates stacked real and imaginary components of the synthesis window. This can then be implemented using a transposed 1D-convolutional layer, where the stride parameter is equal to the hopsize, which allows to calculate gradients of a TF-domain network through back-propagation from a time-domain loss. See [161] for details. For the following experiments, a weighted combination of the MSE in the complex domain ($\mathcal{L}_{cMSE}$), the time-domain ℓ_1-loss ($\mathcal{L}_{\ell_1}$) and the weighted SDR ($\mathcal{L}_{wSDR}$) is jointly minimized, resulting in the used overall loss function

$$\mathcal{L}_{tot} = 0.1\,\mathcal{L}_{cMSE} + 0.2\,\mathcal{L}_{\ell_1} + 0.7\,\mathcal{L}_{wSDR}. \tag{4.5}$$

The weighted SDR loss has been defined in [157] and is calculated as

$$\mathcal{L}_{wSDR}(x, s, \hat{s}) = -\beta\frac{\langle s,\, \hat{s}\rangle}{\|s\| \cdot \|\hat{s}\|} - (1-\beta)\frac{\langle x-s,\, x-\hat{s}\rangle}{\|x-s\| \cdot \|x-\hat{s}\|} \tag{4.6}$$

where β is the energy ratio between the clean source and the accompaniment $x - s$:

$$\beta = \frac{\|y\|^2}{\|y\|^2 + \|x-y\|^2} \tag{4.7}$$

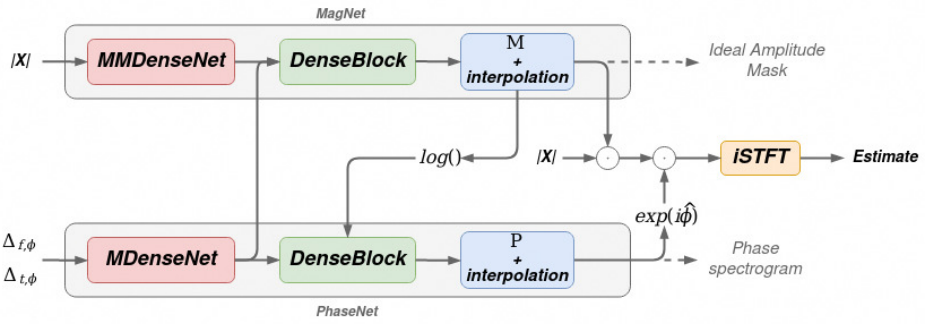


Figure 4.9: Illustration of the MagNet-PhaseNet approach in order to exploit and estimate phase information.

Before the network is trained on this composite loss, it seems reasonable to bootstrap the individual magnitude and phase networks: first, the MagNet can be trained on directly estimating the ideal amplitude mask without any information from the PhaseNet, using a standard MSE loss. In order to prevent the PhaseNet from solely relying on information coming from the MagNet, it is first pre-trained on minimizing the MSE on the target phase spectrogram only using the input features $\Delta_{t,\phi}$ and $\Delta_{f,\phi}$, before it is subsequently re-trained to include the estimated source magnitude of the (fixed) MagNet. After this bootstrapping process, the whole system may be jointly optimized on $\mathcal{L}_{tot}$.

Tab. 4.11 sums up the results. It shows the baseline performance, the MagNet trained on the ideal amplitude mask using the classification formulation, a MagNet-PhaseNet model trained directly on minimizing $\mathcal{L}_{tot}$, one that follows the bootstrapping procedure but is only trained on the complex MSE in the TF-domain, and another bootstrapped model that is subsequently optimized on the $\mathcal{L}_{tot}$. Unfortunately, the MagNet-PhaseNet approach performs rather poorly compared to the baseline. Without a bootstrapping process, the model does not even outperform the baseline. The results also indicate that the PhaseNet actually hurts the performance, since the combined approach trained on the $\mathcal{L}_{cMSE}$ is actually less accurate than the MagNet alone. This may be attributed to the very small model size of the PhaseNet, however, increasing it would counteract the intention behind the design and decrease the system's efficiency. Additionally, although the bootstrapped model trained on the $\mathcal{L}_{tot}$ achieves a slightly better performance, it is surprising that this performance increase is most prominently reflected in the interference reduction (which is roughly 1.5 dB higher), rather than in the SDR. At this point, it is important to note that the experiments with regard to time-domain objectives took an extremely long time to compute, which was probably caused by a computational bottleneck in the data-loading pipeline in combination with a disadvantageous cluster filesystem. While the planned number of epochs for re-training on the time-domain loss was 70 epochs, the training had to be manually killed after 30. Subsequently, the MagNet-PhaseNet approach has not been pursued any further. In general, it seems not very usable within the context of this thesis. Nevertheless, it is interesting from a theoretical and technical viewpoint, and further experiments with better tuning may be able to use this framework in a more successful way.

Table 4.11: Results for experiments investigating the use of exploiting and estimating phase information.

Model	SDR	SIR	SAR
Baseline	5.46	11.71	**5.87**
MagNet, IAM	5.19	11.25	5.72
$\mathcal{L}_{tot}$, no bootstrapping	5.44	11.37	5.78
$\mathcal{L}_{cMSE}$, bootstrapped,	5.06	11.45	5.38
$\mathcal{L}_{tot}$, bootstrapped	**5.49**	**12.93**	5.47

4.6 Deep clustering

This section discusses the utilization of a deep clustering objective [108] in order to improve upon the baseline model. Complying to the concept of binary masking, the deep clustering framework assumes that each TF-bin of the mixture is exclusively associated with a certain source. But instead of estimating the target directly, DNNs are used to generate high-dimensional features or *embeddings* for each TF-bin. The embedding space ideally offers a representation that is convenient for the clustering of the TF-bins with respect to the partitions, or rather sources, they are associated with. After the spectral clusters have been found (e.g. via the simple and

efficient k-means algorithm), a binary mask may be constructed in order to separate the sources. Therefore, the deep clustering objective is to minimize the distance between embeddings of bins belonging to the same source, while maximizing the distance between embeddings for elements of different sources. The outputs of the network are K-dimensional vectors for each TF-bin, where K denotes the dimension of the embedding space, and the vectors are normalized to have *unit-length*. These are gathered in an embedding matrix $\boldsymbol{V} \in \mathbb{R}^{TF \times K}$. Likewise, $\boldsymbol{Y} \in \mathbb{R}^{TF \times N}$ is a label matrix that holds one-hot encoded row vectors indicating which of the N sources dominates the corresponding TF-bin. With the help of the estimated and the ideal *affinity* matrix denoted by $\boldsymbol{V}\boldsymbol{V}^T$ and $\boldsymbol{Y}\boldsymbol{Y}^T$, respectively, which indicate the affinity of one TF-bin to all other TF-bins in terms of source labels, the deep clustering objective can then be conveniently formulated as:

$$\mathcal{L}_{DC}(\boldsymbol{V}, \boldsymbol{Y}) = \|\boldsymbol{V}\boldsymbol{V}^T - \boldsymbol{Y}\boldsymbol{Y}^T\|_F^2 = \|\boldsymbol{V}^T\boldsymbol{V}\|_F^2 + \|\boldsymbol{Y}^T\boldsymbol{Y}\|_F^2 - 2\|\boldsymbol{V}^T\boldsymbol{Y}\|_F^2 \tag{4.8}$$

Note that the right hand side of Eq. (4.8) is an equivalent formulation of the clustering loss, yet it is more computationally efficient since it does not require the computation of the entire $TF \times TF$ affinity matrices, which will typically be large.

Alternative cost functions for deep clustering have been proposed in [162], including one that resembles linear discriminant analysis (LDA), for instance. Additionally, [162] recommends to reduce the influence of low-energy TF-bins, which have somewhat arbitrary labeling and have no strong correlation with a certain source, by adding a weight matrix $\boldsymbol{W}$, with weights corresponding to magnitude ratios of the mixture $\boldsymbol{X}$ at a certain TF-bin indexed by t, f over the sum of the magnitudes over all bins $w_{t,f} = \frac{|\boldsymbol{X}_{t,f}|}{\sum_{t,f} \boldsymbol{X}_{t,f}}$, yielding the weighted cluster loss $\mathcal{L}_{wDC}$:

$$\mathcal{L}_{wDC}(\boldsymbol{V}, \boldsymbol{Y}) = \|\boldsymbol{W}^{\frac{1}{2}}(\boldsymbol{V}\boldsymbol{V}^T - \boldsymbol{Y}\boldsymbol{Y}^T)\boldsymbol{W}^{\frac{1}{2}}\|_F^2 \tag{4.9}$$

In practice, $\mathcal{L}_{wDC}$ can be calculated as efficiently as the standard cluster loss, by simply scaling the elements of $\boldsymbol{V}$ and $\boldsymbol{Y}$ with the corresponding weights $w_{t,f}$ before applying Eq. (4.8).

The deep clustering framework has proven to be effective and flexible. For instance, an interesting approach has been proposed in [163], which jointly represents all music sources in an embedding space and uses an auxiliary network conditioned on the one-hot encoding of the sources for generating the parameters of a Gaussian mixture model (GMM) that parameterizes the embedding space for each source. Mask estimation can then be achieved by estimating the posterior distribution over the classes given the network embeddings. The auxiliary network is trained along with the original one using the Expectation-Maximization (EM) algorithm. While this is a promising approach to have a single multi-source network, the joint modeling of all sources seems inefficient with respect to the aim of this thesis.

One particularly popular extension is the Deep Attractor Network [164]. It tries to incorporate the deep cluster approach while at the same time remaining conditioned on a mask inference objective by calculating masks with the help of so-called attractors $\boldsymbol{A} \in \mathbb{R}^{K \times N}$, which can be thought of as source centroids in the embedding space. During training, the attractor points are calculated as

$$a_{k,n} = \frac{\sum_{tf} v_{tf,k} \cdot y_{tf,n}}{\sum_{tf} y_{tf,n}} \tag{4.10}$$

and can subsequently be used to obtain a mask

$$\boldsymbol{M}_n^{Attr.} = softmax(\boldsymbol{V} \cdot \boldsymbol{A}) \tag{4.11}$$

which enables the "end-to-end" training on the actual mask-estimation loss. This matching on the actual desired separation objective has been shown to increase the quality of the results on

speech separation tasks, but certainly also holds for music sources. For instance, [165] made use of deep attractor networks for joint separation and activity detection of music sources.

Another approach that explores the use of the deep clustering loss for music source separation has been performed in [110]. It uses a multi-task training objective with two separate network output sections or "heads", hence it is referred to as the *chimera* architecture. While one output section is trained on mask-inference, the other head is used to generate the embeddings for a clustering objective. The second output head only acts as a regularizer for the network and can be dropped during training, an approach that is similar to what has been discussed in Section 4.3. The combination of the weighted clustering loss with a standard mean-squared error $\mathcal{L}_{MSE}$ on the spectrogram- or mask-inference head then simply amounts to a weighted sum of both terms

$$\mathcal{L}_{Chim}^{wDC} = \alpha\,\mathcal{L}_{wDC} + (1-\alpha)\,\mathcal{L}_{MSE} \tag{4.12}$$

with $0 \leq \alpha \leq 1$. The chimera method has shown to consistently improve the separation performance. It has also been adopted for experiments conducted in [162], which showed that the mask-inference head should not directly branch off from the embedding layer and that the network benefits from a split of the outputs at an earlier hidden layer.

4.6.1 Results

The following experiments investigated the use of the chimera framework applied to the MM-DenseNet. The concept uses two output heads, similar to the multi-task training setup, but in this case one head is used for spectrogram approximation while the second head is used to minimize the weighted cluster loss $\mathcal{L}_{wDC}$ instead of accompaniment estimation. The embeddings have a dimension of $K = 20$ and are calculated using a single composite-function-layer (BatchNorm + ReLU + 2D-conv) with 20 convolutional kernels of size 3×3. Of course, the number of input channels for this layer depends on the location of the split for the two heads. Again, it is possible to branch off the second network output right after the encoder, but this did not yield good results in the multi-task experiments and also does not comply with [110] or [162]. Thus, only two splitting locations are considered in a preliminary experiment: a split before the *last densely connected block* (similar to Fig. 4.4), or before the actual output layer of the MMDenseNet, the very *last convolutional layer*. Tab. 4.12 shows the results for $\mathcal{L}_{Chim}^{wDC}$ with a weight of $\alpha = 0.2$. Unsurprisingly, the tight coupling of magnitude estimation with the embeddings in the post-DenseBlock scenario reduce the expressiveness of the model with regard to the actual desired goal. By contrast, introducing the auxiliary regularization mechanism of the deep clustering objective in an earlier hidden layer before the output densely connected block of the MMDenseNet seems to be a working strategy.

In order to match the spectrogram approximation (SA) and the deep clustering objective, it seems reasonable to use an attractor loss with mask estimation as in Eq. (4.11) in order to learn a suitable embedding space. The cluster loss is then effectively formulated using a MSE between the estimated masked mixture and the target magnitude. Similarly, another strategy is to estimate masks instead of spectrograms. Since, the deep clustering loss takes inspiration from binary masking schemes, using a phase-sensitive mask as the second target could cause a mismatch and impede learning. The results, given in Tab. 4.13, investigate the use of WF-masks (WF-MA) instead. It is clear from the results that the regularizing effect of the deep clustering objective is able to improve the overall performance of the MMDenseNet measured in SDR, but even more significantly helps the interference reduction. This regularizing effect turns out to be relatively small in the case of the attractor-style loss function, which likely stems from the condition of both heads on similar training objectives. The chimera approach works particularly well for the WF-MA configuration. Though the training objectives in this

Table 4.12: Results of the chimera approach for different splitting locations of the output sections ($\alpha = 0.2$).

Split	SDR	SIR	SAR
post-DenseBlock	5.28	11.20	5.77
pre-DenseBlock	**5.58**	**11.65**	**6.03**

Table 4.13: Results for different (pre-DenseBlock) configurations of the chimera approach, with different values for the weight α.

Method	α	SDR	SIR	SAR
Baseline	-	5.46	11.71	5.87
SA	0.2	5.58	11.65	6.03
SA, Attractor loss	0.2	5.43	11.44	5.89
WF-MA	0.2	5.63	13.05	5.79
WF-MA	0.5	5.82	13.81	5.69
WF-MA	0.8	5.68	14.12	5.54

case are also somewhat similar (both heads are able to obtain a mask), the network is able to benefit from the explicit formulation of the auxiliary clustering loss using affinity matrices in the embedding space. An interesting observation can be made when the relative importance of both objectives is considered: as the value of α increases and more weight is given to the clustering loss, the interference reduction significantly increases while the SAR decreases. This effect can be attributed to the binary masking properties of the clustering objective, which introduces characteristic musical noise. A value of $\alpha = 0.5$ seems to be a good sweet-spot. To demonstrate the effectiveness of the clustering loss, Fig. 4.10 shows a principal component analysis of the embeddings for a batch of samples taken from test set.

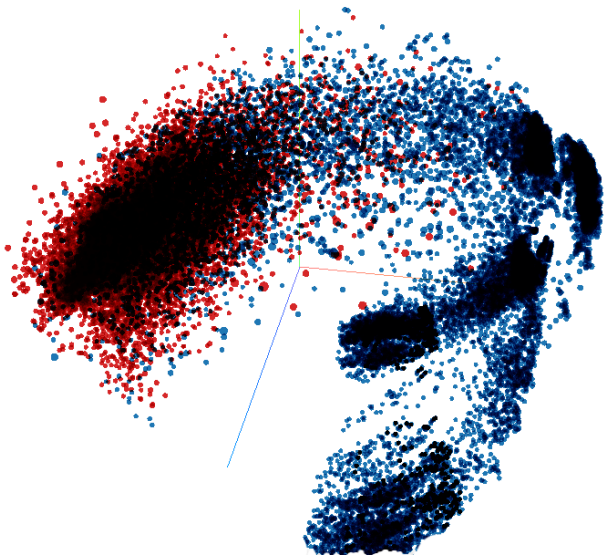


Figure 4.10: Plot of embeddings calculated from a batch of mixture signals of the musdb18 test set, depicted using the first 3 principal components. The embeddings associated with TF-bins of the vocal source are displayed in red, those associated with the accompaniment in blue.

4.7 Summary

This section gives a concise recap of the results presented in this chapter. The evaluation metrics for the best-performing model of each of the discussed approaches are summarized in Tab. 4.14.

The performance of the best-performing models for each approach are summarized in Tab. 4.14. Initial experiments were concerned with details related to the baseline architecture, the MMDenseNet. The use of the rather small kernel size of 3×3 for the spatial convolutions has been validated in the context of music signals, both in terms of performance as well as efficiency. Also, constraining the range of the output to positive values with an additional ReLU non-linearity yields more interpretable results and slightly better performance due to the constrained search space. Furthermore, different scalings of input features and training targets on the performance have been investigated in Sections 4.1.4 and 4.1.5. Specifically, it has been shown that logarithmically-scaled target magnitude spectrogram destabilize training, which can be explained by the increased impact of irrelevant (logarithmically scaled) TF-bins on the loss and the network's finite accuracy. The effect of different scalings (logarithmic and mean-variance normalization) of input features is less sensitive on the performance, however, simple linear magnitude spectrograms have been found to work best.

Section 4.2 discusses the benefits of mask estimation and shows that further constraining the search space via mask-based objectives facilitates training and obtains better separation quality. In particular, learned masks trained directly on the source spectrogram can be advantageous over explicitly formulated and possibly sub-optimal ones, but this only applies if the MMDenseNet has already been pre-trained on mask estimation.

The value of multi-task training objectives has been investigated in Section 4.3. The motivation for these experiments was to teach the network a consistent representation of the audio material that is convenient for separation and induce regularization through joint modeling of the singing-voice and accompaniment sources. Additionally, estimating the complementary sources in parallel allows for different mask computation approaches. However, significant performance gains over the single-task procedure were missed, and could only be achieved with slight decreases in efficiency and an explicit Wiener filter-style mask formulation with the additional source during inference time.

Two alternative designs that aim to improve the crude multi-band approach of the MMDenseNet have been proposed in Section 4.4, transferring the processing of different frequency bands into channels and dense blocks, respectively. While the multi-band dense blocks enable an effective and flexible framework with enhanced information flow for multi-band processing in contrast to the multi-band channel approach which destroys the spectral correlation between bands, extensive experiments showed contrary results. In fact, upon inspection of kernel weights of the multi-band channel configuration, it may be concluded that band-limited kernel sharing for distinct pattern modeling may not be overly important in terms of separation performance. Moreover, additional cost-intensive experiments are necessary to do a fair comparison and proper evaluation of the different models. For these reasons and the slightly disappointing results for the multi-band designs both in terms of performance and efficiency, multi-band processing approaches have not been further explored. It should nevertheless be pointed out that, while the MMDenseNet represents a good trade-off between performance and efficiency, further experiments with the multi-band dense framework might be able to yield more sound and superior models, especially considering the rather high and hard cutoff at 8 kHz and the restricted information sharing of the baseline architecture.

So far, phase information has been entirely ignored with respect to both input and output of the network. Section 4.5 discusses ways to address this issue, since it appears to be a very limiting factor. Problems with the cyclical nature of the phase can be bypassed by leveraging the instantaneous frequency and group delay of the phase. These have been used to construct useful

input features, which can be fed to a parallel network that classifies discretized values of the phase for each TF-bin and interpolates between them in order to obtain a continuous estimate. Since magnitude and phases are directly related via their gradients, the respective magnitude and phase estimation networks are also designed to exchange intermediate features. The resulting complex short-time spectrogram are transformed to the waveform domain, which allows to jointly train the networks e.g. on a performance-based metric. However, this approach only minimally improves the performance, which neither justifies the rather tedious bootstrapping process nor the increased computational demands of the phase estimation. The results indicate that the used phase estimation network has been constructed too small. Though better results may be obtained by increasing the model size (apart from longer training routines), this is not compatible with the efficiency objective of this thesis.

Finally, a clustering objective with magnitude-ratio weighting is employed as an auxiliary task during training of the MMDenseNet, similar to the Chimera++ approach [110], [162], which has proven to work very well, especially in combination with Wiener Filter mask estimation. Since considerations relating to the efficiency objective presented in the next chapter will introduce changes with regard to the architecture as well as the feature representation, a combination of the mask-based and deep clustering approaches, which have been identified as beneficial in terms of performance and may also be conveniently used in combination, has not been discussed thus far, and will be explored later on in Section 5.2.2.

Table 4.14: Summary of the findings relating to performance improvements. Median BSSEval scores in dB over the test set for the best performing models of each method are presented.

Model	SDR	SIR	SAR
Baseline	5.46	11.71	5.87
Learned mask, PSA-mask pre-training	5.63	11.17	**6.45**
Joint vocal and accomp., joint mask comp.	5.63	12.37	5.92
Multi-band channels	5.53	11.59	5.90
Multi-band dense blocks (MBDB-1)	5.44	11.23	5.98
MagNet + PhaseNet, $\mathcal{L}_{tot}$, bootstrapped	5.49	12.93	5.47
Deep clustering, $\alpha = 0.5$	**5.82**	**13.81**	5.69

5 Efficiency

The previous chapter addressed several methods regarding performance improvements, and established a solid foundation of a source separation system. Continuing from the previous insights, this chapter pursues a seemingly contrary aim by investigating strategies to decrease inference times as well as model size and increase the DNN's overall resource efficiency. Naturally, we strive for solutions that find a good trade-off between both aspects and are able to offer efficient inference without sacrificing the separation quality. Because the resulting system should ideally be applicable in real-time scenarios, the practical consequences of algorithmic latency requirements are discussed first. Subsequently, the effects of compressed feature map representations and structural efficiency through effective use of bottleneck layers and a pruning method are discussed in Section 5.2, Section 5.3 and Section 5.4, respectively.

5.1 Dealing with a small context size

The baseline model is part of the family of convolutional neural networks which share weights to compute activations only locally and are thus much more lightweight than other neural network architectures. Despite the calculation of local features, CNNs are able to capture and model long-term dependencies because the effective receptive field of the feature maps grows with the number of layers, as depicted in Fig. 5.1. However, CNNs are stateless in the sense that no information is retained over successive forward passes. Consequently, the degree to which temporal information may be maximally exploited depends on the given input. Since audio data, and music in particular, exhibits strong temporal dependencies, the target source can only be accurately described and modeled if the context size of the input to the CNN is sufficiently large. For this reason, CNNs for audio tasks typically receive a large context window, which is limited in length only by practical considerations such as GPU memory. For instance, the effective context size of the MMDenseLSTM [86] is roughly 8.3 seconds.

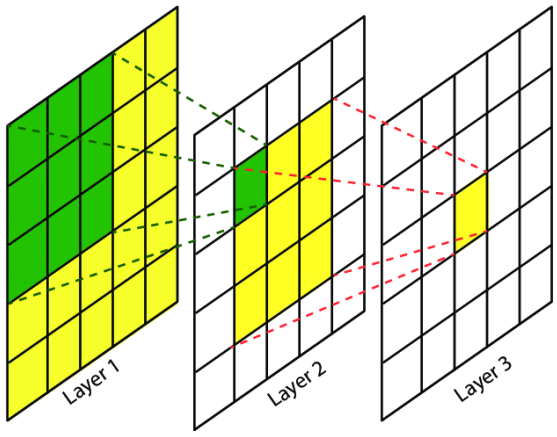


Figure 5.1: Receptive fields of 3 successive convolutional layers for a kernel size of 3×3 [166]. One element in the third layer has access to information of a 5×5 receptive field in the first layer (yellow).

However, when time-critical processing is required, which is frequently the case for audio applications, the algorithmic latency has to be kept adequately low. In on-line processing environments the input sequence to a CNN can essentially be constructed in two ways. One straightforward way is to wait for the context size buffer to fill up, perform the inference pass, and write out all processed samples at once. Assuming that the model is able to execute the forward pass in a reasonable amount of time - less than the context buffer length or an external callback period - the algorithmic latency is exactly equal to the context size.

The alternative approach tries to retain a large context size and yet keep the latency low by performing an inference pass for smaller sub-sequences of the buffer and writing out only the corresponding amount of newly processed samples. In theory, the context window may then be made arbitrarily long by aggregating context over time and re-using samples from the past. This algorithm may be operated at the lower latency bound, which is determined by the STFT frame-length for systems based in the TF-domain. In this case, an inference pass is done for every new incoming frame - which also implies that the compute load will be higher by a factor equal to the context size. This is of course the extreme case and designs using larger sub-sequences of frames may be more appropriate in practice. Also, dilated convolutions (as used e.g. in [167]) offer great potential in handling larger context sizes and receptive fields in an efficient manner. Additionally, instead of only retaining input frames to keep the context size large, parts of network *activations* may also be saved, shifted on the time axis, and re-applied for the processing of the next sub-sequence. Thus, only the parts of the feature maps that actually benefit from the new input would be calculated. However, an implementation of this approach could turn out to be very difficult to realize in practice and may result in rather messy solutions. But in general, this method of using larger context sizes is only feasible if the network is already resource-efficient enough and can offer a sufficient throughput.

Another and much more natural way to capture long-term temporal dependencies is to make use of hidden states through additional recurrent units and thus relax the dependency on a sufficiently long context window. The MMDenseLSTM [86] demonstrates how to efficiently construct such a hybrid network from the MMDenseNet architecture. However, as already stated in Section 3.1, recurrent layers are much more compute intensive and parameter-heavy, and are also frequently coupled with fully connected layers.

As it hard to assess a-priori whether either of the last two methods can be made efficient enough at all for real-world scenarios, simply working with a reduced context size seems to be a more "reliable" way towards efficiency for now. The reduction of the feature map sizes along the time axis will directly translate to faster inference times and lower memory footprints, even if the efficiency of the algorithm itself has not changed of course. Keeping in mind that other strategies may still be investigated at a later stage, lowering the context size seems to be a viable path if the reduced temporal information at the input does not cause disproportionally large degradations in separation quality.

5.1.1 Baseline results

The maximum allowed algorithmic latency is application dependent, but should stay as low as possible in order to maintain an appropriate overall response time of the application. In the case of Plug-Ins for digital audio workstations, delay times of $100 - 200$ milliseconds are considered as rather large, but still acceptable. In order to have consistent down- and up-sampling in combination with concatenation from skip-connections to the encoder to the decoder of the MMDenseNet, it is convenient to have input spectrograms that are powers of 2 in size, on both the time- and the frequency-axis. Using an FFT-length of $L = 2048$ samples, a hop-size of $H = 512$ samples and a sampling rate of $R = 32\,\text{kHz}$, these requirements are fulfilled with a context size $C = 8$ frames, which corresponds to $\frac{(C-1)*H+L}{R} = 176\,\text{ms}$.

Table 5.1: Results using the baseline system configuration for various context sizes used in training and during testing.

Context size				
Training	Testing	SDR	SIR	SAR
64	64	5.46	11.71	5.87
64	16	4.57	11.15	5.58
64	8	4.24	8.75	5.41
8	8	4.93	10.05	5.61

Although CNNs can handle inputs of varying size, drops in performance may be expected if the context size used in the inference stage is smaller than the context size used during training. Convolutional kernels of deeper layers are trained to rely on and produce features of a certain receptive field size. If the input is smaller than the expected context size, the model simply reaches the boundaries of what can be deduced from the data. Tab. 5.1 shows that avoiding such a mismatch of context sizes is indeed highly beneficial for the separation performance: therefore, a model trained directly using shorter context performs better in this scenario. More importantly though, Tab. 5.1 demonstrates that, despite the exceptionally small effective context size of 176 ms - orders of magnitudes less than commonly used CNN context lengths - the model is still able to successfully perform source separation with a loss of only 0.53 dB in SDR and roughly 1.6 dB in interference reduction. However, it has to be acknowledged, that the small context size seems to evoke a slight tremolo effect, with fluctuating amplitudes in the resulting vocal estimates. However, since the effect is not excessively prominent, it has not been further dealt with.

A summary of the effect of different context sizes with regard to performance and efficiency is given in Tab. 5.2. It shows the number of multiply-accumulate operations (MACs) that the network performs during the execution of a forward pass for a single input sample. This measure roughly correlates with the expected inference time. The model's performance is summed up in the SDR, given in dB. Additionally, the respective relative changes are stated, denoted by Δ. Since the model is the same for both experiments and only the spectrogram sizes change, the number of parameters is the same for both models. Rather unsurprisingly, the number of MACs is directly proportional to the feature map size: in light of the tremendous reduction of MACs by a factor of 8, the loss in objective separation quality appears to be quite acceptable.

Table 5.2: The effects of different input context sizes on separation quality (SDR given in dB) and the number of Multiply-Accumulate operations (MACs) needed to perform a forward pass for a single sample input. Δ denotes the relative change in percent. The number of parameters is the same for both models.

	MACs		SDR	
Context size	Value	Δ	Value	Δ
64	4.168×10^9	-	5.46	-
8	0.521×10^9	$-87.5\,\%$	4.93	$-9.7\,\%$

5.2 Mel-scaled representations

Since reduced feature map sizes directly translate to shortened inference times, a compression of the feature representation along the frequency axis, in addition to the decreased context sizes along the time axis, is clearly a sensible strategy in order to increase the computational efficiency of the algorithm. Using a different frequency scaling also allows to dismiss the linear frequency resolution of Fourier magnitude spectra, which does not correlate well with the human perception of pitch. Thus, the perceptually motivated Mel-scale seems to be particularly suited for these purposes. The input of the mixture is run through a Mel filter bank before being fed to the network input, and the DNN outputs a Mel-scaled estimate.

However, the compression introduced by the Mel filters is a lossy procedure. Fully operating the network at the Mel scale raises the question of how the output can be converted in order to recover a magnitude spectrogram estimate suited for inversion with the iSTFT. Of course it is possible to learn an appropriate inverse filter jointly with the network weights during optimization (the same actually also applies to the input transform). But a much more straightforward and efficient way is to use the filter weights for linear interpolation of the frequency bins with the corresponding Mel bins. While direct interpolation of Mel-spectrogram estimates will likely fail in producing high-quality results, this method should work reasonably well for the reconstruction of spectral masks. Thus, instead of converting source magnitude spectrograms directly, using a Mel-scaled mask for reconstruction and applying it to the original input mixture allows to retain fine-grained information of the original linear frequency resolution. If the Mel filter bank is formulated as a matrix of filter weights $\boldsymbol{F}_{Mel} \in \mathbb{R}^{P \times L}$ with P denoting the number of filters used and L denoting the number of Fourier components, a Mel spectrogram input can be easily obtained from the STFT mixture magnitude spectrogram $|\boldsymbol{X}|$ as

$$|\boldsymbol{X}_{Mel}| = \boldsymbol{F}_{Mel} \cdot |\boldsymbol{X}| \,. \tag{5.1}$$

The reconstruction of a mask $\hat{\boldsymbol{M}}$ of the original TF-representation from Mel-scaled masks $\hat{\boldsymbol{M}}_{Mel}$ using linear interpolation therefore has an analogous formulation

$$\hat{\boldsymbol{M}} = \boldsymbol{F}_{Mel}^{T} \cdot \hat{\boldsymbol{M}}_{Mel} \tag{5.2}$$

where $\boldsymbol{F}_{Mel}^{T}$ is simply the transpose of the Mel filter bank $\boldsymbol{F}_{Mel}$. The implementation of the *librosa* toolbox [168] is used to calculate the Mel filter bank. It uses the formula proposed in [169] to convert from Hertz to Mel, with a linear scale below and a logarithmic scale above a corner frequency of 1 kHz. The filters have triangular shape with uniformly spaced center frequencies on the Mel scale, and are depicted in Fig. 5.2. They are not normalized, so that each filter aims for a peak value of 1, which enables convenient linear interpolation.

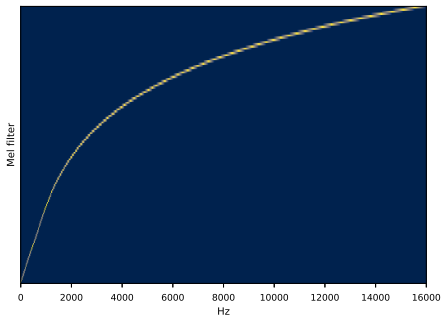


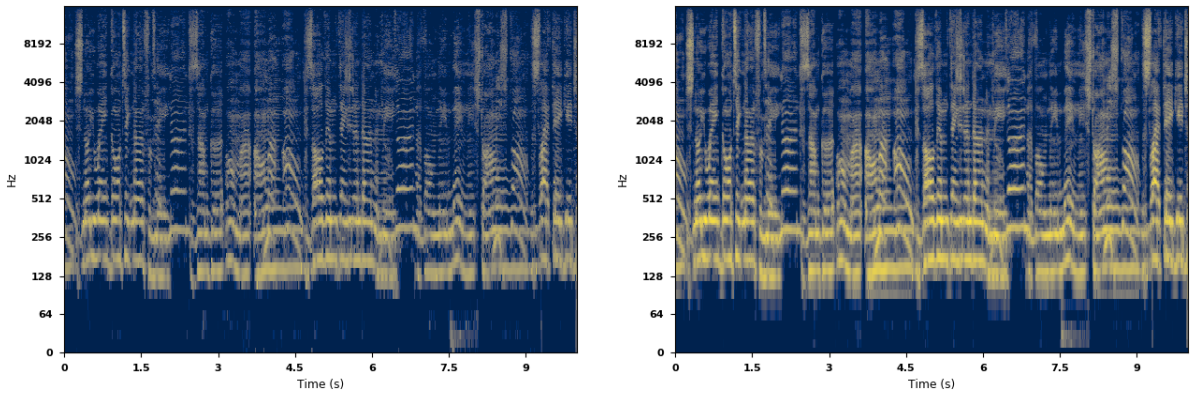
Figure 5.2: The used 128 triangular Mel filters, each aiming for a target value of 1, displayed over the linear frequency axis given in Hz.

5.2.1 Preliminary experiments using oracle masks

In order to evaluate the effect of employing Mel-scaled mask estimates on performance, a preliminary experiment using oracle masks is conducted first. Wiener filter-style masks are obtained from both linear magnitude spectrograms and Mel-spectrograms of the true target sources, calculated per track over the whole training set according to Eq. (2.13) with a value of $\alpha = 2$ for the exponent. Note that due to the collapse of multiple bins along the frequency axis into single values and the absence of a Mel-scaled phase estimate, the use of phase-sensitive masks is not recommended in this case. Tab. 5.3 shows the resulting upper bounds in performance for different Mel filter bank sizes. The oracle experiment proves that linearly interpolated 128-bin Mel-scaled masks are able to produce estimates of satisfactory separation quality, while gaining a theoretical increase in efficiency by a factor of 8. Although more aggressive compression of the frequency scale is conceivable as well, the performance gradually decreases with smaller numbers of Mel bins. Since a reduction by a factor of 8 should yield good performance in combination with significant speed-ups, using 128 Mel filters seems to be an adequate choice for further experiments. The effect of the recovered Mel-scaled mask is represented graphically in Fig. 5.3. Although the Mel-mask introduces slight smearing effects along the frequency axis of the spectrogram, the general structure of the two estimates is very much the same.

Table 5.3: Oracle performance of Mel-scaled Wiener filter-like masks reconstructed using interpolation to linear-scaled masks. Values correspond to the median value in dB over the whole musdb18 training set (100 songs).

Mask type	SDR	SIR	SAR
STFT, 1024 bins	9.74	20.97	9.90
Mel, 128 bins	8.41	18.65	8.75
Mel, 96 bins	7.94	17.93	8.35
Mel, 64 bins	7.14	16.77	7.68



(a) Using linear STFT-mask

(b) Using 128-bin Mel-mask

Figure 5.3: Log-Magnitude spectrograms of oracle source estimates, obtained via a linear-scaled STFT mask (a) and a Mel-mask of 128 bins (b) applied to a 10-second of a 10-second mixture extract.

5.2.2 Results

The use of the Mel-scale questions the efficacy of the baseline model's multi-band structure. Naturally, the Mel filters introduce a blur along the frequency axis. Additionally, it also takes into account the roughly logarithmic perception of frequencies of the auditory system. Though this does not change the fact that different frequency bands exhibit different spectral patterns, the Mel-scale might help to reduce the differences between such patterns, since music is composed for the human ear. Hence, a full-band MDenseNet-based model might be better suited for this task. In order to enable a fair comparison between the two models, the number of parameters has been adjusted to be approximately equal. For this reason, the full-band model features growth rates of $k = 18$ and the number of layers within each densely connected block is $L = 4$. All other parameters are the same for both models. While the multi-band input is originally split at a frequency of 8 kHz, the cutoff frequency for the 2-band input of the Mel-MMDenseNet has been lowered to 2 kHz, since the balance between the individual sub-networks would otherwise be broken as the MDenseNet associated with the higher frequency band would otherwise receive a very small number of Mel-bins compared to the other one. Since the 2 kHz split cuts the Mel-spectrogram directly in two halves, this method again complies with the original linear scale version in this regard.

The shorter context size of 8 frames is used for both training and testing in all following experiments. Thus, input and output dimension of the network are of size 128×8. Since the Mel-scaled output should represent a spectral mask for subsequent linear interpolation and multiplication with the original linear STFT mixture magnitude, both networks are trained on Wiener filter-like masks computed from the true target Mel-spectrograms.

Tab. 5.4 shows the performance of the different model configurations using Mel-scaled inputs and outputs in comparison to the standard approach using STFTs. With a drop in overall SDR performance of roughly only 0.2 dB, the results seem to encourage the usage of Mel-scaled representations for efficient source separation. Moreover, Tab. 5.4 seems to indicate that networks operating on Mel-scaled features are even able to achieve better interference reduction. However, this effect most likely stems from the mask-based training with Wiener filter targets, which generally obtains higher SIRs compared to direct spectrogram approximation as used for the baseline system. Additionally, the full-band model indeed exhibits a slight advantage in performance compared to the multi-band model.

Table 5.4: Performance of multi- and full-band networks with Mel-scaled inputs and a context size of 8 frames, trained to estimate Wiener filter-style masks computed from Mel-spectrograms. The output Mel-mask is linearly interpolated with the Mel filter weights to create a linear scale mask, which is applied to the original mixture to obtain the source estimate.

Model	SDR	SIR	SAR
MMDenseNet	4.93	10.05	5.61
Mel-MMDenseNet	4.70	11.22	4.87
Mel-MDenseNet	4.73	11.37	5.05

Leveraging an auxiliary deep clustering objective

It has already been established in Chapter 4 that, in addition to a Wiener filter mask approximation objective, the weighted deep clustering loss may be utilized as an effective regularization scheme. By simply formulating the affinity matrices with the corresponding Mel-scaled representations, the method can straightforwardly be applied in this context. Just as in the linear-scale case, the additional network output section is branched off before the last densely connected

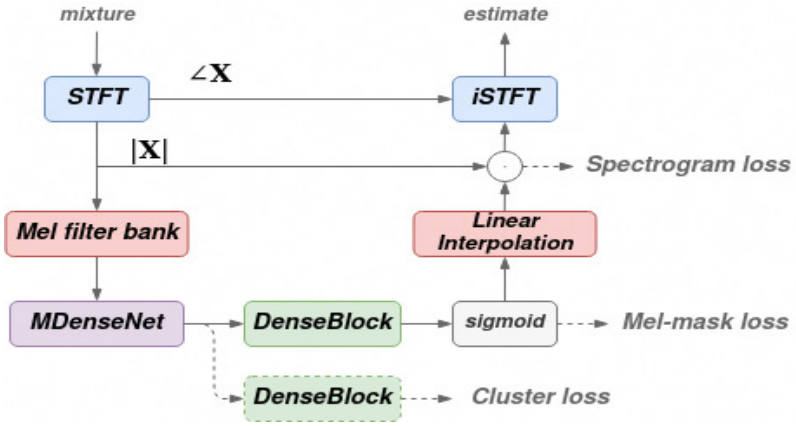


Figure 5.4: Overview of the used system for Mel-mask estimation with linear interpolation and the deep clustering objective. Parts denoted by dashed lines are used during training only.

block and the two objectives equally contribute with a factor of 0.5 to the overall loss, which has shown favorable results in Section 4.6. The overall system is illustrated in Fig. 5.4. The results, obtained using the advantageous full-band Mel-MDenseNet, are summarized in Tab. 5.5. Additionally, the effect of a re-training on the true magnitude spectrograms with regard to performance is investigated: instead of optimizing the Wiener filter mask directly, we can use Eq. (5.2) to obtain the linear-scaled mask, directly apply it to the original mixture magnitude, and use the resulting estimate to calculate a spectrogram approximation loss. This way, the network should learn to generate a Mel-scaled mask that is amenable to linear interpolation and separation on the linear scale, without explicitly specifying how the mask should look like. The re-training may optionally include a deep clustering objective as well. For the results shown in Tab. 5.5, the re-training procedure has been conducted for 20 additional epochs.

The auxiliary deep clustering objective significantly improves the separation performance, also in the case of Mel-scaled representations. In particular, all 3 evaluation metrics experience an improvement compared to the Mel-MDenseNet trained on a single-task mask optimization task. Surprisingly, a re-training using the true linear-scaled source spectrograms decreases the performance, despite the better conceptual match with the actual evaluation objective. Re-training schemes that also include the deep cluster loss seem to work better than only minimizing the MSE to the true source. This indicates that the condition of the original training objectives is strong, and that the re-training leads the models out of an already quite good minimum. Naturally, longer re-training might be able to solve this issue, but would render the original training unnecessary. For this reason, this has not been investigated any further in the context of this thesis. Due to the deficient effect of re-training and the significantly higher SIR of the originally trained model, further experiments in the next sections build upon the latter approach.

Table 5.5: Performance of different full-band Mel-MDenseNet configurations trained to jointly minimize the deep clustering loss (DC) in addition to WF-style Mel-mask approximation (WF-MA) with equal weights. The models are optionally retrained on minimizing the Euclidian distance to the corresponding linear-scale source magnitudes (SA).

Target	Re-Training	SDR	SIR	SAR
WF-MA	-	4.73	11.37	5.05
WF-MA + DC	-	**4.87**	**11.68**	5.12
WF-MA + DC	SA	4.79	9.81	5.77
WF-MA + DC	0.8 SA + 0.2 DC	4.84	9.90	5.75
WF-MA + DC	0.5 SA + 0.5 DC	**4.87**	10.27	**5.61**

Efficiency

The benefit of exploiting and inferring Mel-scaled spectrograms and masks, respectively, is summarized in Tab. 5.6. While the theoretical increase in inference-time efficiency by a factor of 8 is not met - due to the slight architectural changes for the full-band Mel-MDenseNet - the reduction of total multiply-accumulate operations is substantial. Additionally, the full-band Mel-MDenseNet uses a similar yet less number of parameters compared to the multi-band model, and thus facilitates a slightly lighter memory footprint. Furthermore, this increase in efficiency comes only with a very small cost in performance, since the use of the deep clustering objective keeps the relative loss of $-1.22\,\%$ in SDR with regard to the baseline favorably small.

Table 5.6: The effects of Mel-scaled representations on separation quality (SDR given in dB) and the number of Multiply-Accumulate operations (MACs) needed to perform a forward pass for a single sample input. Δ denotes the relative change in percent.

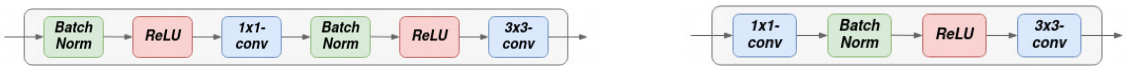
	MACs		Parameters		SDR	
Model	Value	Δ	Value	Δ	Value	Δ
Baseline	521.012×10^6	-	255.481×10^3	-	4.93	-
Mel-MDenseNet + DC	108.862×10^6	$-79.11\,\%$	238.517×10^3	$-6,64\,\%$	4.87	$-1.22\,\%$

5.3 Structural efficiency through point-wise convolutions

A particularity of DenseNets is the linear growth of the number of feature maps (along the channel axis) with the number of layers in the densely connected blocks. Although the dense connectivity increases the information flow and allows to keep the number of kernels per layer k small compared to other architectures, given that the number of outputs per layer typically increases the number of inputs, the successively increasing feature map sizes appear to be computationally ineffective. The authors of the original DenseNet paper [129] address this issue with point-wise convolutions. These convolutional layers of kernel size 1×1, also called *bottleneck layers*, precede each of the layers in a dense block and can be used to reduce the number input feature maps flowing into the layer. This kind of bottleneck compression allows for substantial improvements with regard to efficiency, and thus seems like a natural extension to the baseline MMDenseNet which does not leverage the technique.

As pointed out in [170], residual architectures - and DenseNets in particular - greatly benefit from a *pre-activation structure*, meaning that the normalization and activation layer (in this case BatchNorm and ReLU) precede the convolutional layer. This way the composite layer may normalize its input individually. This is important, because the result of a concatenation of feature maps coming from different previous layers is likely to exhibit large variations in its distribution. Therefore, the typical bottleneck compressed layer as used in [129] has the structure depicted in Fig. 5.5a, with an additional normalization and activation layer.

For efficient inference, BatchNorm layers can usually be folded into convolutional layers if the normalization follows after the convolution. However, pre-activation structures prevent such a fusion of operations, since folding the first normalization layer of a composite function into the 3x3 convolution of the previous one would break the pre-activation advantage for all succeeding densely connected layers. The structure illustrated in Fig. 5.5b addresses this issue by simply removing the additional BatchNorm and ReLU operations introduced by the bottleneck compression, rendering the composite function to a sequence of Conv (1x1) - BN - ReLU - Conv (3x3). Since the point-wise convolutions only learn to compress the feature maps and the composite unit primarily remains a pre-activation structure, the increase in efficiency caused



(a) *(b)*

Figure 5.5: Two different composite functions of a densely connected block with bottleneck compression and pre-activation structures. Structure (b) allows to fuse the BatchNorm layer into the point-wise convolution for efficient inference.

by the elimination of BatchNorm layers should outweigh the effect of dropping the additional layers. Additionally, it is possible to include the BatchNorm layer at the output of the composite function and therefore apply the normalization *before* the feature maps are sent to subsequent layers. However, since initial results without the additional output normalization proved to work quite well, this method has not been investigated since the bias terms also introduce additional operations. At this point it is worth recalling that the beneficial effect of BatchNorm does not necessarily stem from its ability to reduce internal covariate shift, but rather from a smoother loss landscape and hence more stable gradient directions [140]. The single normalization layer in Fig. 5.5b might be sufficient for this purpose.

Although the MMDenseNet does not make use of point-wise convolutions in densely connected blocks, it does utilize them at transition layers, namely the average pooling layers. This compression technique is also used in the original DenseNet, but, as pointed out in [171], this kind of feature map reduction hurts performance. Since the activations already experience a "spatial" reduction along the TF-axes, the additional compression along the channel dimension seems to overcharge the change in representation. Interestingly though, the MMDenseNet does not actually use these layers for compression, but keeps the number of feature maps before and after the 1x1 convolutions the same. The contribution of these layers is uncertain, even if no compression is applied, which is why the point-wise convolutions in transition layers may be dropped entirely.

5.3.1 Results

The experiments presented in this section build upon the best previous model, which is trained using the chimera-style deep clustering objective (Mel-MDenseNet-DC). First, the effect of discarding the point-wise convolutions in pooling layers is assessed. As shown in Tab. 5.7 the model, termed Mel-MDenseNet-DC-NoDSConvs, is indeed able to obtain a slightly better performance with regard to SDR and SAR, while removing spurious parameters.

Next, the use of bottleneck compression inside densely connected blocks is investigated. Interestingly, the amount of compression in [129] is set to a fixed value of 4 times the growth rate of the following convolutional layer, which in many cases would actually correspond to an expansion instead of a reduction of feature maps, and accordingly increase the computational cost. The model termed Mel-MDenseNet-DC-NoDSConvs-BC reduces all incoming feature maps by a factor of 2 and adheres to the structure shown in Fig. 5.5b. Despite the significant reduction of feature maps, the model essentially maintains the separation performance.

Tab. 5.8 reveals the tremendously positive effects of using proper bottleneck compression on computational efficiency. While the number of operations as well as the model size can be reduced by a factor of more than 1.4, the separation quality measured in SDR is not affected at all by bottleneck layers, and even increases due to the removal of spurious point-wise convolutions in the downsampling path.

Table 5.7: Results for experiments investigating the effect of point-wise convolutions and bottleneck compression on the performance.

Model	SDR	SIR	SAR
Mel-MDensenet-DC	4.87	11.68	5.12
Mel-MDenseNet-DC-NoDSConvs	4.96	11.51	5.30
Mel-MDenseNet-DC-NoDSConvs-BC	4.91	11.51	5.22

Table 5.8: The effect of bottleneck compression on the number of model parameters, separation quality (SDR given in dB) and the number of Multiply-Accumulate operations (MACs) needed to perform a forward pass for a single sample input. Δ denotes the relative change in percent.

	MACs		Parameters		SDR	
Model	Value	Δ	Value	Δ	Value	Δ
Mel-MDenseNet-DC	108.862×10^6	-	238.517×10^3	-	4.87	-
Mel-MDenseNet-DC-NoDSConvs-BC	76.677×10^6	$-29.57\,\%$	169.095×10^3	$-29,11\,\%$	4.91	$+0.82\,\%$

5.4 Structured Pruning

Resource-efficiency for Deep Neural Networks has emerged into one of the most important and most actively addressed research topics among the deep learning community, and essentially evolves around three different principal yet non-disjoint directions of research. Naturally, a strategy that strives to improve a networks efficiency ideally encompasses concepts from all of these.

Structural efficiency Of course, the structure of the DNN architecture should preferably pursue an inherently efficient design. The previous Section 5.3 addresses considerations relating to structural efficiency in a fairly straightforward way through the use of point-wise convolutions in order to compress the number of feature maps that have to be processed onto a more compact representation. For convolutional neural networks in particular, a large number of design choices has been devised in this vein. Prominent representatives of this group are for instance the works of [172], [173] and [174], which are based on factorizations of the convolutional operator. Another branch in this field of research is associated with automated machine learning (AutoML) [175], which has brought forward very exciting findings recently, e.g. [176]. Automatic search routines can be successful in obtaining very efficient architectures, usually if the search space is constrained accordingly to meet certain related criteria. Popular neural architecture search procedures have been proposed in [177] and [178], for instance.

Quantization Since smaller bit-widths are able to yield significant savings in model size and memory footprint, and speed-up computations, quantization is also considered to play a crucial role for the deployment of DNNs in resource-constrained environments. Many different strategies have been developed in order to perform quantization for deep neural networks. For example, network weights may be quantized with or without their corresponding activations, using fixed or adaptive codebooks during or after training. Further particularities include e.g. the sensitivity of BatchNorm layers to coarse quantization. An overview of methods in this field is given in [179]. Interestingly, it has been shown that it is possible to maintain high accuracy even in spite of extreme quantization onto binary weights - a survey of recent methods employing binary weights and/or activations is given in [180].

Weight pruning Whereas quantization methods leverage the redundancy in the number of bits, weight pruning leverages redundancy in the number of network parameters. Modern deep neural networks are usually heavily over-parameterized, which also seems to be the reason for their excellent learning abilities. In general, the aim of pruning techniques is to identify some sub-network within the parameter space of the model that performs comparably or equally well as the original one (see e.g. [181]), by evaluating the importance of the network weights according to a certain relevance metric. One simple yet effective metric is for example the magnitude of the corresponding weights, but a variety of different pruning methods has been explored in the literature [182], [183], [184]. As demonstrated for example in [185], it is certainly possible to combine quantization and pruning methods, also on highly structural efficient backbones [186] and may even be optimized jointly in a unified framework [187], [188], [189]. Nevertheless, while quantization significantly reduces the computational complexity of DNNs, pruning methods allow to spare expendable computations entirely and are thus prioritized in the context of this thesis. In addition, compression techniques are commonly evaluated on (image) classification tasks and it is not completely clear how well the discussed methods, and quantization methods in particular, translate to tasks that are required to retain high-resolution features such as music source separation algorithms.

Determining convenient and hardware-friendly sparsity structures

Although the fine-grained, unstructured pruning of weight connections achieves impressive compression rates, several practical drawbacks are often ignored: reported memory savings typically do not include space requirements of additional indices, load imbalance may lead to poor performance on parallel processors and randomness in memory access patterns caused by unstructured sparsity may prevent successful caching mechanisms from working effectively. These shortcomings can be avoided by constraining pruning methods to well-defined, hardware-friendly sparsity structures. Thus, as far as real-world settings are concerned, structured pruning methods should be preferred for resource-efficient inference. [190] even discourages the use of unstructured pruning altogether, since specialized hardware does not seem to benefit from these methods either.

2D-convolutions (also called spatial convolutions) are usually implemented as dense matrix multiplications, as shown in Fig. 5.6 and described in more detail in [191]. Without resorting to specialized sparse matrix multiplications, which are neither well supported by deep learning libraries nor hardware accelerators, the choice of sparsity structures that allow for actual inference speed-ups and convenient implementation is rather limited and corresponds to very coarse-grained structures. Apart from pruning whole layers, sparsity structures that map conveniently to hardware and benefit the execution of dense matrix multiplications are essentially input channels, output channels and filter "columns", meaning the columns of the flattened weight tensor $\boldsymbol{F}_m$ shown in Fig. 5.6. In combination with the removal of the corresponding row in the reshaped data matrix $\boldsymbol{D}_m$, pruning these columns results in direct efficiency increases. However, the accompanying implementation of this approach has to be performed using the low-level matrix representation of the convolutional operation, which might pose difficulties with regard to different frameworks and target environments. While the pruning of single convolutional kernels (output channels, i.e. the rows of the flattened convolutional weight tensor) may be performed on a higher level, a change in the number of output channels of a layer also causes a change in the number of input channels for subsequent layers, and thus also calls for painstaking implementation procedures, especially with residual architectures. This leaves input channels as hardware-friendly sparsity structures, which can be straightforwardly implemented in combination with a removal of the corresponding input feature map. This method seems to be a bit ineffective though, because each layer produces a certain amount of feature maps which might not even be leveraged by subsequent layers. However, keeping in mind that the target architecture of concern is based on the DenseNet and encourages heavy feature re-use, this method seems to be adequate. .

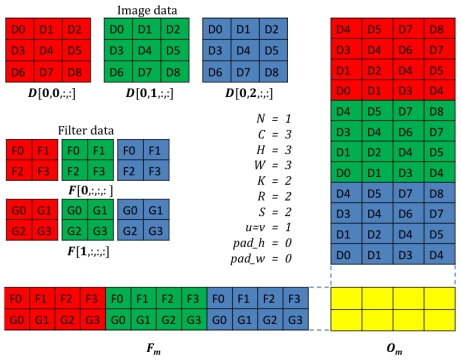


Figure 5.6: Lowering of a spatial 2D-convolutional operation to a dense matrix multiplication [191]. The input is a single 3×3 $(H \times W)$ "RGB-image" with $C = 3$ channels and a batch size of $N = 1$. The convolutional layer consists of $K = 2$ output kernels, of size 2×2 $(R \times S)$. The operation is reformulated as a matrix multiplication of the flattened convolutional layer $\boldsymbol{F}_m \in \mathbb{R}^{K \times CRS}$ and a re-arranged data matrix $\boldsymbol{D}_m \in \mathbb{R}^{CRS \times NPQ}$, where P and Q denote height and width of the target output tensor $\boldsymbol{O} \in \mathbb{R}^{N \times K \times P \times Q}$ which has the intermediate representation $\boldsymbol{O}_m \in \mathbb{R}^{K \times NPQ}$.

5.4.1 Parameterized Structured Pruning

However, jointly pruning whole weight structures of a deep neural network is more sensitive to accuracy degradations than the unstructured pruning of individual connections. In this thesis, the effect of the recently introduced Parameterized Structured Pruning is investigated [192]. It is a regularization-based pruning method that explicitly parameterizes a certain sparsity structure j, and jointly optimizes so-called structure parameters α_j together with the network weights. The relevance of structure j then corresponds to the magnitude of its assigned structure parameter, $|\alpha_j|$. A penalty - such as a ℓ_1 regularization or weight decay - is used to push the parameters towards zero, enabling a clear distinction between important and unimportant structures.

In order to learn the parameters together with the 4-dimensional weight tensors $\boldsymbol{W}$ of convolutional layers, the sub-tensors $\boldsymbol{w}_j$ that make up structure j are substituted by the tensor $\mathbf{q}_j = \mathbf{w}_j \nu_j$ during the forward pass, where ν_j denotes the associated *sparse* structure parameters. These are obtained by evaluating the *dense* structure parameters α_j with the help of a threshold function

$$\nu_j(\alpha_j) = \begin{cases} 0 & |\alpha_j| < \epsilon \\ \alpha_j & |\alpha_j| \geq \epsilon \end{cases} \tag{5.3}$$

where ϵ corresponds to a tunable pruning threshold. Since the threshold function is not differentiable at $\pm\epsilon$ and the gradient of the sparse parameters ν_j is zero on the pruning interval $[-\epsilon, \epsilon]$, their gradient is defined using a straight-through-estimator [193]

$$\frac{\partial E}{\partial \nu_j} = \frac{\partial E}{\partial \alpha_j} = \sum_{m=1}^{M} \frac{\partial E}{\partial w_{j,m}}\,, \tag{5.4}$$

where $w_{j,m}$ corresponds to the m^{th} weight of structure j, which has M weights in total. E represents the objective function. Thus, α_j is updated in accordance with the predominant direction of the weights. Accordingly, in the case of channel input pruning, the gradient of the parameters can be expressed as follows:

$$\frac{\partial E}{\partial \alpha_c} = \sum_{k=1}^{K} \sum_{r=1}^{R} \sum_{s=1}^{S} \frac{\partial E}{\partial W_{k,c,r,s}} \tag{5.5}$$

Since the dense instead of the sparse parameters are updated, improperly pruned structures are allowed to reappear during training, which yields a more stable training behaviour. Weight decay is employed to push the weights below the threshold while taking their magnitude into account. Although ℓ_1-regularization may seem more natural for sparsity induction as it decays parameters exactly to zero, weight decay has been found to yield clearer distinctions between important and unimportant structures. For efficient inference, the additional parameters may simply be fused into the weight structure.

5.4.2 Results

Parameterized structured pruning (PSP) is performed on the input channels of all convolutional kernels (point-wise as well as 3x3 convolutional layers), except for the potentially sensitive output and input layers. The structure parameters are initialized using a uniform distribution in the interval [-1, 1], since the Gaussian distribution proposed in the original paper lead to convergence problems and effectively prevented the network from learning. The value of 1×10^{-5} for decaying the weight tensors is the same as in previous experiments, however, the penalty for all α_j has been increased to 1×10^{-2} in order to ensure sufficient regularization of the sparsity-inducing structure parameters. Since the training is conducted with the Adam optimizer, it is important to decouple the weight decay from the optimization steps [133]. Also note that while PSP has been originally developed for standard stochastic gradient descent, the adaptive learning rates should not interfere with the general pruning algorithm, since the gradients for the structure parameters and the weights are equivalent. The sparsity threshold ϵ is set to zero initially and is gradually increased when training has saturated. After each threshold increase, the network is fine-tuned for a short training round of 2 epochs. The experiments are based on the best previous model, termed Mel-MDenseNet-DC-NoDSConvs-BC, which uses bottleneck compression.

In order to assess the effect of the used pruning technique, Fig. 5.7 presents the SDR as an overall performance measure and the amount of induced sparsity in percent, measured as the proportion of pruned to non-pruned parameterized weight structures, as a function of the pruning threshold ϵ. The plot clearly shows the inverse relationships between these two objectives. Exemplary results for different pruning thresholds are displayed in Tab. 5.9.

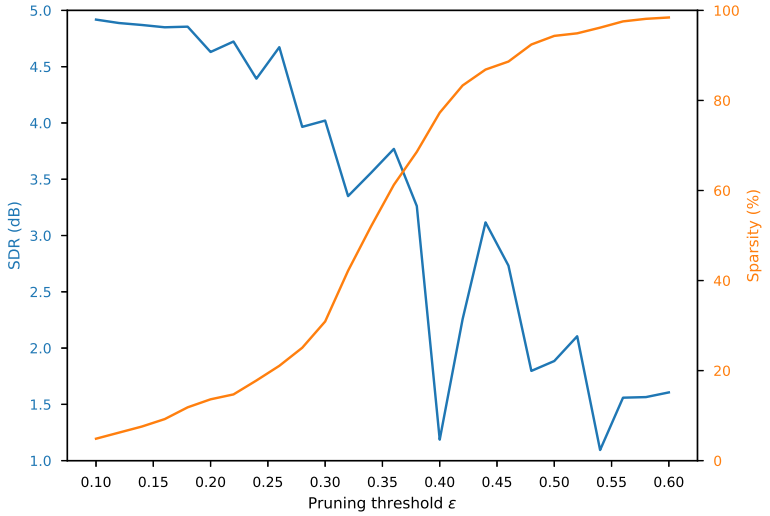


Figure 5.7: Overall performance measured in the median SDR in dB over the test set and the amount of induced sparsity across all parameterized weight sub-tensors in percent for various pruning thresholds ϵ.

Unfortunately, the performance starts to degrade quite early and collapses before reaching a sparseness of around 15 %, at a pruning threshold of roughly $\epsilon = 0.2$. This rather poor behaviour may be attributed to several aspects. First, music source separation is a high-resolution regression task and the network will thus be in general much more susceptible to small changes of the weights compared to classification tasks that arguably measure more global and coarse features. Secondly, the combination of bottleneck compression for structural efficiency and PSP is sub-optimal, since the former method significantly limits the effectiveness of the latter one: as bottleneck compression is based on a significant reduction of feature maps, an additional pruning of input channels for convolutional kernels is very likely to constrain the network's learning capabilities too much. Applying the pruning method on the original computational units might even be able to further reduce the number of operations and parameters. While assessing this hypothesis would be interesting, this experiment has been left open to future investigations. Moreover, further experiments would be necessary in order to thoroughly evaluate the impact of different optimization routines on PSP. While the individual learning rate of the used adaptive optimization procedure should not affect the relation between the structure parameters and the corresponding weights (since they receive the same gradients), the pruning technique might still be more robust with standard stochastic gradient descent. In addition, though the PSP paper [192] considers modern and large CNN architectures that make heavy use of batch normalization, the interaction between weight decay and BatchNorm is rather intricate [142], [143], [144] which might also lead to difficulties for this regularization-based pruning technique.

Nevertheless, it should be pointed out that the pruning method is certainly successful in the removal of at least some of the structure, without hurting the performance too much. Tab. 5.10 shows the efficiency improvements due to pruning with a threshold of $\epsilon = 0.18$, which yields a reasonable balance between the overall performance and the amount of induced sparsity. Compared to the prior model configuration, a relative improvement of roughly 6.5 % and 5.2 % for the number of multiply-accumulate operations and the number of parameters, respectively, certainly justifies the relative loss in performance of only 1.02 %.

Table 5.9: Results for experiments investigating the effect of parameterized pruning on the performance. Sparsity denotes the amount of pruned to non-pruned parameterized structures.

Model	SDR	SIR	SAR	Sparsity
Mel-MDenseNet-DC-NoDSConvs-BC	4.91	11.51	5.22	-
... + PSP $\epsilon = 0.14$	4.87	11.69	5.13	7.57 %
... + PSP $\epsilon = 0.18$	4.86	11.22	5.14	11.53 %
... + PSP $\epsilon = 0.22$	4.72	9.78	5.73	17.75 %

Table 5.10: The effect of Parameterized Structured Pruning (PSP) ($\epsilon = 0.18$) on the number of model parameters, separation quality (SDR given in dB) and the number of Multiply-Accumulate operations (MACs) needed to perform a forward pass for a single sample input. Δ denotes the relative change in percent.

	MACs		Parameters		SDR	
Model	Value	Δ	Value	Δ	Value	Δ
Mel-MDenseNet-DC-NoDSConvs-BC	76.677×10^6	-	169.095×10^3	-	4.91	-
... + PSP $\epsilon = 0.18$	71.680×10^6	−6.52 %	160.228×10^3	−5.24 %	4.86	−1.02 %

5.5 Summary

The findings of this chapter are summarized in Tab. 5.11. As the main goal of the thesis is to obtain a system that ideally suits real-time, resource-constrained online inference in a block-processing fashion, a discussion of the accompanying limitations and appropriate methods for this purpose is presented first. In particular, the system is required to handle reduced input context windows. Initial results showed that, despite a very small context size of 8 frames, the system is capable of separating music with relatively little degradation in separation quality. If the network is efficient enough, recurrent units can always be added at a later stage. Similarly, the context may always be augmented with past input frames, to improve the temporal modeling abilities of the convolutional architecture. For these reasons, the thesis proceeds by considering a small context scenario of 176 ms.

As reduced feature map sizes directly correlate with increased computational efficiency, Section 5.2 deals with the compression of feature representations along the frequency axis. The perceptually motivated Mel-scale has proven to be expedient for this purpose. The network infers a Mel-scaled mask from a Mel input magnitude spectrogram, from which the linear-scaled mask can subsequently be reconstructed via simple linear interpolation with the filter weights. While the separation performance stays largely unaffected, this approach naturally has a tremendously positive impact on the computational efficiency, The utilization of the Mel scale also questions the multi-band structure of the model, and a comparable full-band architecture of roughly the same number of parameters does indeed show slight advantages over the original one. Additionally, analogous to the linear scale case, an auxiliary deep clustering objective is formulated from the true Mel-spectrogram sources and successfully leveraged to refine the generalization and thus performance of the network. Rather surprisingly though, the incorporation of the linear mask reconstruction as part of the optimization procedure and the direct re-training on the spectrogram approximation objective were not able to improve upon the explicit Wiener Filter-like mask estimation.

Section 5.3 addresses the structural efficiency of the network by incorporating bottleneck layers in densely connected blocks in order to compress intermediate feature maps along the channel direction while maintaining an efficient pre-activation structure. In combination with a removal of redundant convolutions in pooling layers, the system is able to preserve the separation performance, while significantly decreasing the compute load, measured as the number of forward-pass operations and parameters of the network, by a factor of approximately 1.4.

Finally, Section 5.4 considers the parameterized pruning of weight structures that conveniently map to hardware, such as input channels of convolutional kernels. While the obtained compression ratio is rather small - which may be attributed to the nature of the high-resolution regression task, the already reinforced compactness of the model and the sub-optimal interaction with the explicit bottleneck compression in particular - the pruning technique positively contributes to the system's efficiency.

Table 5.11: Summary of the findings relating to efficiency improvements. Median BSSEval scores in dB over the test set as well as the number of multiply-accumulate operations and the number of parameters for the best performing models of each method are presented.

Model	SDR	SIR	SAR	MACs	Parameters
Baseline, 64 frames	5.46	11.71	5.87	4.168×10^9	255.481×10^3
Baseline, 8 frames	4.93	10.05	5.61	0.521×10^9	255.481×10^3
Mel-MDenseNet-DC	4.87	11.68	5.12	108.862×10^6	238.517×10^3
Mel-MDenseNet-DC-NoDSConvs-BC	4.91	11.51	5.22	76.677×10^6	169.095×10^3
... + PSP, $\epsilon = 0.18$	4.86	11.22	5.14	71.680×10^6	160.228×10^3

5.5.1 Real-time capability of the approach

In order to assess the real-time capability of the resulting model, the Mel-MDenseNet-DC-NoDSConvs-BC pruned with a threshold of $\epsilon = 0.18$, the average execution time of a forward pass on the Intel Core i7-8700B 6-core CPU of a Mac Mini 8.1 ("Late 2018") is measured. The PyTorch framework [130] allows for convenient serialization of the model graph using the TorchScript Just-in-time compiler. This serialized representation of the network can subsequently be exported into the open exchange neural network format *ONNX* [194] for interfacing with various frameworks on a wide range of platforms. Additionally, this offers the usage of a highly optimized run-time library [195] for efficient inference, which has been utilized in combination with the Intel *DNNL* hardware accelerator [196] for the measurements. *perf_counter_ns* of the built-in Python library *time* has been used to obtain a measurement of the single-batch forward pass execution time with a context size of 8 frames (176 ms at a sampling rate of 32 kHz) over the full musdb18 test set.

On average, the model takes about 4 milliseconds to complete the forward pass. This is an effective improvement by a factor of 6.5 over the baseline, which takes about 26 milliseconds to process a single example with reduced context size. While it has to be acknowledged that the measurement platform is a rather modern one, this is well within the range of typical audio buffer block-sizes, and thus validates the real-time capability of the proposed approach.

6 Conclusion

This final chapter sums up the previous findings and points out the usefulness as well as the caveats of the proposed approach.

In general, the described approach towards a real-time capable, resource-friendly deep neural network for music source separation yields a reasonable trade-off between separation performance and computational efficiency. A key strategy that allowed to accomplish this goal was the reduction of feature map sizes, which substantially decreased the computational requirements. In particular, operating the algorithm on the Mel-scale proved to work extremely well. While the subjective impression with the corresponding reconstructed linearly-scaled masks may slightly change, the effect on the separation quality is deemed insignificant. Although the reduced input context size is of course a major limiting factor, it admittedly also plays a crucial role for the fast run-time performance of the system, without actually contributing to the efficiency of the algorithm. Nevertheless, it is interesting that the performance of the convolutional network can be kept sufficiently high, in spite of the little temporal information it receives as input.

Most of the discussed methods that strive to improve the system's performance unfortunately lack remarkable outcomes. Only rather small changes in the overall performance, indicated by the SDR, have been attained, with maximum differences in the range of $\pm 0.5\,\mathrm{dB}$. However, the beneficial effect of the deep clustering framework [110], [162] has proven to be practical and reliable, as it is applicable to convolutional, U-Net-based architectures such as the MMDenseNet, and results in zero computational overhead during inference time. Additionally, though the results with regard to (fine-tuned) learned masks confirm the advantage of input-to-output skip-filtering connections for denoising auto-encoders [92], in combination with Mel-scaled outputs and a deep clustering approach, the direct Wiener Filter estimation tends to work slightly better in practice.

The results also somewhat discourage the utilization of phase information for inference in resource-constrained environments. In practice, purely magnitude based processing tends to be simpler and more efficient, and the plain application of the mixture phase for signal reconstruction from the TF-domain is still backed by state-of-the-art source separation systems [86], [83], [89], [90]. To this regard, rather than an estimation of the target phase via cost-intensive DNN-based algorithms, in certain circumstances it might be more appropriate to use efficient phase reconstruction techniques, e.g. [197], as a post-processing step. However, it has to be pointed out that the phase-estimation network considered in this thesis has been of deliberately small scale and was difficult to train.

Moreover, a successful re-design of the architecture for multi-band processing has not been attained, especially also with regard to the efficiency. The MMDenseNet seems to be a rather robust architecture, and serves as a good source separation backbone throughout most conducted experiments. Although the results question the effectiveness of the multi-band processing and the eventually used model employs a full-band structure, further experiments with the proposed multi-band dense blocks may still be helpful in finding better designs.

6.1 Outlook

As already stated above, the reduced context size represents a major limiting factor for convolutional architectures since it determines their temporal modeling abilities. It also introduces slight tremolo effects for the models developed in this thesis. The inclusion of uni-directional additional recurrent units similar to [86], for example efficient variants of gated recurrent units (GRUs), may be very well suited to alleviate this problem. Additionally, the use of dilated convolutions may also be investigated in order to leverage increased context sizes in an efficient way. Furthermore, an investigation of the increasingly popular attention-based neural networks, which recently have been very successfully applied to music source separation [89], [90], would be very interesting in this context, especially with regard to the efficiency objective.

Apart from methods for improved temporal modeling, future work will probably want to focus on increasing the efficiency of the model even further, since a more efficient system also allows for larger context sizes, which in turn might deliver greatly improved overall performance. Using point-wise convolutions for bottleneck compression of feature maps is a known technique to improve computational efficiency, and helps to effectively reduce the computational load for the DenseNet-based architecture. However, this method is only moderately compatible with subsequent pruning algorithms, which are expected to yield higher compression rates as they induce sparsity in a more adaptive and flexible way. To this end, efficiency improvements might be obtained by discarding bottleneck compression in favor of releasing the full potential of pruning algorithms. Additionally, one might also be interested in exploring alternative (structured) pruning methods for this purpose, e.g. [198], or considering more fine-grained sparsity structures.

Still, maintaining a structural efficient architecture should not be ignored. In particular, the memory footprint of the model has largely been omitted in this work. Since the number of feature maps grows linearly with the number of layers in a densely connected block, the size of all intermediate feature maps is order of magnitudes larger than the model itself, especially for high-resolution regression tasks like audio source separation. The DenseNet requires a lot of data caching and, despite its parameter efficiency, consumes a considerable amount of energy, as shown in [199]. This also directly affects inference times, since limited DRAM bandwidths may decrease the system's efficiency, which is especially problematic for GPU processing. [200] addresses this issues with the DenseNet and provides a more carefully designed dense architecture design, which should be very beneficial in this context. Additionally, a re-design may draw more heavily from the idea of factorized convolutional operations (e.g. [172]) in order to increase structural efficiency.

Less memory consumption may also be achieved through quantization methods, which should also lead to significant improvements in resource-efficiency. While the use of binary neural networks will probably not be applicable to the task of music source separation, quantization aware training procedures for 8-bit integers may be able to preserve performance while significantly shrinking the computational requirements of an ideally already efficient model.

Finally, it has to be noted that the used evaluation metrics are only partly suited to represent the separation quality of the estimated sources obtained using the proposed approach, and that subjective listening tests are necessary in order to properly evaluate the source separation system.

Bibliography

[1] J. S. Park, "Unsupervised approach to music source separation using generalized dirichlet prior," Ph.D. dissertation, Seoul National University, 2018.

[2] K. D. Martin, "Sound-source recognition: A theory and computational model," Ph.D. dissertation, EECS, MIT, Cambridge, MA, USA, 1999.

[3] E. L. Saldanha and J. F. Corso, "Timbre cues and the identification of musical instruments," *The Journal of the Acoustical Society of America*, vol. 36, 1964. [Online]. Available: https://doi.org/10.1121/1.1919317

[4] M. Chasin, "Music and hearing aids — an introduction," *Trends in Amplification*, 2012. [Online]. Available: https://doi.org/10.1177/1084713812468512

[5] E. Cano, D. FitzGerald, and K. Brandenburg, "Evaluation of quality of sound source separation algorithms: Human perception vs quantitative metrics," in *European Signal Processing Conference (EUSIPCO)*, 2016.

[6] D. Ward, H. Wierstorf, R. Mason, E. Grais, and M. Plumbley, "Bss eval or peass? predicting the perception of singing-voice separation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[7] E. Cano, J. Liebetrau, D. Fitzgerald, and K. Brandenburg, "The dimensions of perceptual quality of sound source separation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 04 2018, pp. 601–605.

[8] E. M. Grais, H. Wierstorf, D. Ward, R. Mason, and M. D. Plumbley, "Referenceless performance evaluation of audio source separation using deep neural networks," *CoRR*, vol. abs/1811.00454, 2018. [Online]. Available: http://arxiv.org/abs/1811.00454

[9] M. Cartwright, B. Pardo, and G. Mysore, "Crowdsourced pairwise-comparison for source separation evaluation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 04 2018.

[10] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, "Subjective and objective quality assessment of audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, 2011.

[11] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, July 2006.

[12] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR - half-baked or well done?" *CoRR*, vol. abs/1811.02508, 2018. [Online]. Available: http://arxiv.org/abs/1811.02508

[13] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 Signal Separation Evaluation Campaign," *arXiv e-prints*, p. arXiv:1804.06267, Apr 2018.

[14] E. Vincent. (2007) Stereo audio source separation evaluation campaign - evaluation criteria. [Online]. Available: https://www.irisa.fr/metiss/SASSEC07/?show=criteria

[15] J. C. Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[16] C. Schörkhuber and A. Klapuri, "Constant-q transform toolbox for music processing," in *7th Sound and Music Computing Conference, Barcelona, Spain*, 2010, pp. 3–64.

[17] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[18] T. Gerkmann and E. Vincent, *Spectral Masking and Filtering*. John Wiley & sons, Ltd., 2018, ch. 5, pp. 65–85. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119279860.ch5

[19] A. Liutkus and R. Badeau, "Generalized Wiener filtering with fractional power spectrograms," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 266–270.

[20] D. Fourer and G. Peeters, "Single-channel blind source separation for singing voice detection: A comparative study," *CoRR*, vol. abs/1805.01201, 2018. [Online]. Available: http://arxiv.org/abs/1805.01201

[21] S. Voran, "The selection of spectral magnitude exponents for separating two sources is dominated by phase distribution not magnitude distribution," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 279–283.

[22] D. Wang and J. Lim, "The unimportance of phase in speech enhancement," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 30, no. 4, pp. 679–681, 1982.

[23] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[24] K. Paliwal, K. Wójcicki, and B. Shannon, "The importance of phase in speech enhancement," *speech communication*, vol. 53, no. 4, pp. 465–494, 2011.

[25] P. Mowlaee, J. Kulmer, J. Stahl, and F. Mayer, *Single Channel Phase-aware Signal Processing in Speech Communication: Theory and Practice*. John Riley & Sons, Ltd., 2016.

[26] M. G. Christensen and A. Jakobsson, "Multi-pitch estimation," *Synthesis Lectures on Speech & Audio Processing*, vol. 5, no. 1, pp. 1–160, 2009.

[27] C. Hsu and J. R. Jang, "On the improvement of singing voice separation for monaural recordings using the mir-1k dataset," *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[28] B. Pardo, A. Liutkus, Z. Duan, and G. Richard, *Audio Source Separation and Speech Enhancement*. John Wiley & Sons, Inc., 2018, ch. Applying Source Separation to Music, pp. 345–376.

[29] A. V. Oppenheim, "Speech analysis-synthesis system based on homomorphic filtering," *The Journal of the Acoustical Society of America*, 1969.

[30] Z. Duan, B. Pardo, and L. Daudet, "A novel cepstral representation for timbre modeling of sound sources in polyphonic mixtures," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2014.

[31] Z. Duan, Y. Zhang, C. Zhang, and Z. Shi, "Unsupervised single-channel music source separation by average harmonic structure modeling," *Audio, Speech, and Language Processing, IEEE Transactions on*, 2008.

[32] Y. Meron and K. Hirose, "Separation of singing and piano sounds." in *International Conference on Spoken Language Processing (ICSLP)*, 1998.

[33] E. Cano and C. Cheng, "Melody line detection and source separation in classical saxophone recordings," in *12th International Conference on Digital Audio Effects (DAFx09)*, 2009.

[34] C. J. C. Burges, "Dimension reduction: A guided tour," *Foundations and Trends® in Machine Learning*, vol. 2, no. 4, pp. 275–365, 2010. [Online]. Available: http://dx.doi.org/10.1561/2200000002

[35] J. Shlens, "A tutorial on independent component analysis," *CoRR*, vol. abs/1404.2986, 2014. [Online]. Available: http://arxiv.org/abs/1404.2986

[36] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[37] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2012.

[38] C. Févotte, E. Vincent, and A. Ozerov, *Audio Source Separation*. Springer, 03 2018, ch. Single-Channel Audio Source Separation with NMF: Divergences, Constraints and Algorithms, pp. 1–24.

[39] C. Févotte and J. Idier, "Algorithms for nonnegative matrix factorization with the beta-divergence," *CoRR*, vol. abs/1010.1763, 2010. [Online]. Available: http://arxiv.org/abs/1010.1763

[40] P. Smaragdis and B. Raj, "Shift-invariant probabilistic latent component analysis," *Journal of Machine Learning Research - JMLR*, 01 2008.

[41] C. Févotte and A. Cemgil, "Nonnegative matrix factorizations as probabilistic inference in composite models," *European Signal Processing Conference*, pp. 1913–1917, 01 2009.

[42] D. Fitzgerald, M. Cranitch, and E. Coyle, "On the use of the beta divergence for musical source separation," in *Signals and Systems Conference (ISSC)*, 07 2009, pp. 1 – 6.

[43] E. Vincent and X. Rodet, "Underdetermined source separation with structured source priors," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2004, pp. 327–334.

[44] G. Mysore and P. Smaragdis, "A non-negative approach to semi-supervised separation of speech from noise with the use of temporal dynamics," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 05 2011, pp. 17–20.

[45] J. F. Gemmeke, T. Virtanen, and A. Hurmalainen, "Exemplar-based sparse representations for noise robust automatic speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2067–2080, Sep. 2011.

[46] W. Wang, A. Cichocki, and J. A. Chambers, "A multiplicative algorithm for convolutive non-negative matrix factorization based on squared euclidean distance," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2858–2864, 2009.

[47] J. Durrieu, B. David, and G. Richard, "A musically motivated mid-level representation for pitch estimation and musical audio source separation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1180–1191, Oct 2011.

[48] D. Fitzgerald, M. Cranitch, and E. Coyle, "Extended nonnegative tensor factorisation models for musical sound source separation," *Computational intelligence and neuroscience*, vol. 2008, p. 872425, 02 2008.

[49] A. Ozerov, E. Vincent, and F. Bimbot, "A general flexible framework for the handling of prior information in audio source separation," *IEEE Transactions on Audio Speech and Language Processing*, vol. 20, 05 2012.

[50] C. Joder, F. Weninger, D. Virette, and B. Schuller, "A comparative study on sparsity penalties for nmf-based speech separation: Beyond lp-norms," in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, 10 2013, pp. 858–862.

[51] J. Le Roux, F. J. Weninger, and J. R. Hershey, "Sparse nmf–half-baked or well done?" *Mitsubishi Electric Research Labs (MERL), Cambridge, MA, USA, Tech. Rep., no. TR2015-023*, vol. 11, pp. 13–15, 2015.

[52] A. Lefevre, F. Bach, and C. Févotte, "Itakura-saito nonnegative matrix factorization with group sparsity," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 05 2011.

[53] C. Laroche, M. Kowalski, H. Papadopoulos, and G. Richard, "A structured nonnegative matrix factorization for source separation," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, Aug 2015, pp. 2033–2037.

[54] P. Smaragdis, C. Févotte, G. J. Mysore, N. Mohammadiha, and M. Hoffman, "Static and dynamic source separation using nonnegative factorizations: A unified view," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 66–75, May 2014.

[55] G. Mysore, P. Smaragdis, and B. Raj, "Non-negative hidden markov modeling of audio with application to source separation," in *Proc. 9th Int. Conf. Latent Variable Analysis and Signal Separation*, vol. 6365, 09 2010, pp. 140–148.

[56] P.-S. Huang, S. Chen, P. Smaragdis, and M. Hasegawa-Johnson, "Singing-voice separation from monaural recordings using robust principal component analysis," *Acoustics, Speech, and Signal Processing (ICASSP), International Conference on*, 03 2012.

[57] T. Watanabe, T. Fujisawa, and M. Ikehara, "Vocal separation using improved robust principal component analysis and post-processing," in *2016 IEEE 59th International Midwest Symposium on Circuits and Systems, MWSCAS 2016*, 3 2017.

[58] I.-Y. Jeong and K. Lee, *Latent Variable Analysis and Signal Separation*. Springer, 02 2017, vol. 10169, ch. Singing Voice Separation Using RPCA with Weighted l_1-norm, pp. 553–562.

[59] Z. Rafii, A. Liutkus, and B. Pardo, *Blind Source Separation*. Springer, 2014, ch. REPET for background/foreground separation in audio, pp. 395–411.

[60] J. Foote and S. Uchihashi, "The beat spectrum: a new approach to rhythm analysis," in *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, Aug 2001, pp. 881–884.

[61] D. Fitzgerald, "Harmonic/percussive separation using median filtering," *13th International Conference on Digital Audio Effects (DAFx-10)*, 01 2010.

[62] A. Liutkus, D. Fitzgerald, Z. Rafii, B. Pardo, and L. Daudet, "Kernel Additive Models for Source Separation," *IEEE Transactions on Signal Processing*, vol. 62, pp. 4298–4310, Aug. 2014.

[63] A. Liutkus, D. Fitzgerald, and Z. Rafii, "Scalable audio separation with light kernel additive modelling," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 76–80.

[64] H.-G. Kim and J. Kim, "Music/voice separation based on kernel back-fitting using weighted β-order mmse estimation," *ETRI Journal*, vol. 38, 12 2015.

[65] T. Virtanen, A. Mesaros, and M. Ryynänen, "Combining pitch-based inference and non-negative spectrogram factorization in separating vocals from polyphonic music." in *SAPA@ INTERSPEECH*, 2008, pp. 17–22.

[66] Y. Ikemiya, K. Itoyama, and K. Yoshii, "Singing voice separation and vocal f0 estimation based on mutual combination of robust principal component analysis and subharmonic summation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 2084–2095, 2016.

[67] Z. Rafii, Z. Duan, and B. Pardo, "Combining rhythm-based and pitch-based methods for background and melody separation," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 22, no. 12, p. 1884–1893, Dec. 2014. [Online]. Available: https://doi.org/10.1109/TASLP.2014.2354242

[68] J. Driedger and M. Müller, "Extracting singing voice from music recordings by cascading audio decomposition techniques," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 04 2015, pp. 126–130.

[69] H. Tachibana, Y. Mizuno, N. Ono, and S. Sagayama, "A real-time audio-to-audio karaoke generation system for monaural recordings based on singing voice suppression and key conversion techniques," *Journal of Information Processing*, vol. 24, no. 3, pp. 470–482, 2016.

[70] R. Santos-Rodriguez, T. De Bie, and M. Mcvicar, "Learning to separate vocals from polyphonic mixtures via ensemble methods and structured output prediction," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, ser. ICASSP. Institute of Electrical and Electronics Engineers (IEEE), 5 2016.

[71] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," *Applied and Computational Harmonic Analysis*, vol. 43, no. 2, pp. 233–268, 2017.

[72] B. Hanin, "Universal function approximation by deep neural nets with bounded width and relu activations," *Mathematics*, vol. 7, no. 10, p. 992, 2019.

[73] E. Cano, D. Fitzgerald, A. Liutkus, M. Plumbley, and F.-R. Stöter, "Musical source separation: An introduction," *IEEE Signal Processing Magazine*, vol. 36, 01 2018.

[74] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Joint optimization of masks and deep recurrent neural networks for monaural source separation," *CoRR*, vol. abs/1502.04149, 2015. [Online]. Available: http://arxiv.org/abs/1502.04149

[75] S. Uhlich, F. Giron, and Y. Mitsufuji, "Deep neural network based instrument extraction from music," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 04 2015.

[76] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on DNNs through data augmentation and network blending," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017. [Online]. Available: http://sigport.org/1502

[77] S. I. Mimilakis, K. Drossos, T. Virtanen, and G. Schuller, "A recurrent encoder-decoder approach with skip-filtering connections for monaural singing voice separation," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2017, pp. 1–6.

[78] S. I. Mimilakis, K. Drossos, J. F. Santos, G. Schuller, T. Virtanen, and Y. Bengio, "Monaural singing voice separation with skip-filtering connections and recurrent inference of time-frequency mask," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 721–725.

[79] K. Drossos, S. I. Mimilakis, D. Serdyuk, G. Schuller, T. Virtanen, and Y. Bengio, "Mad twinnet: Masker-denoiser architecture with twin networks for monaural sound source separation," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[80] F.-R. St
"oter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix - a reference implementation for music source separation," *Journal of Open Source Software*, 2019. [Online]. Available: https://doi.org/10.21105/joss.01667

[81] A. Jansson, E. J. Humphrey, N. Montecchio, R. M. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," in *ISMIR*, 2017.

[82] S. Park, T. Kim, K. Lee, and N. Kwak, "Music source separation using stacked hourglass networks," in *International Society for Music Information Retrieval (ISMIR)*, 2018.

[83] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models," Late-Breaking/Demo ISMIR 2019, November 2019, deezer Research.

[84] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," *CoRR*, vol. abs/1706.09588, 2017. [Online]. Available: http://arxiv.org/abs/1706.09588

[85] S. Mobin, B. Cheung, and B. Olshausen, "Convolutional vs. recurrent neural networks for audio source separation," 2018. [Online]. Available: https://openreview.net/forum?id=SkKnhFJPG

[86] N. Takahashi, N. Goswami, and Y. Mitsufuji, "MMDenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation," *CoRR*, vol. abs/1805.02410, 2018. [Online]. Available: http://arxiv.org/abs/1805.02410

[87] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music Source Separation in the Waveform Domain," HAL, Tech. Rep. 02379796v1, 2019.

[88] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[89] T. Li, J. Chen, H. Hou, and M. Li, "Sams-net: A sliced attention-based neural network for music source separation," 2019.

[90] Y. Liu, B. Thoshkahna, A. Milani, and T. Kristjansson, "Voice and accompaniment separation in music using self-attention convolutional neural network," *arXiv preprint arXiv:2003.08954*, 2020.

[91] O. Ronneberger, P. Fischer, and T. 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.

[92] S. Ioannis Mimilakis, K. Drossos, E. Cano, and G. Schuller, "Examining the Mapping Functions of Denoising Autoencoders in Music Source Separation," *arXiv e-prints*, p. arXiv:1904.06157, Apr 2019.

[93] G. Philipp, D. Song, and J. G. Carbonell, "The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions," *CoRR*, vol. abs/1712.05577, 2017. [Online]. Available: http://arxiv.org/abs/1712.05577

[94] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[95] J. Liu and Y. Yang, "Denoising auto-encoder with recurrent skip connections and residual regression for music source separation," *CoRR*, vol. abs/1807.01898, 2018. [Online]. Available: http://arxiv.org/abs/1807.01898

[96] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," *CoRR*, vol. abs/1806.03185, 2018. [Online]. Available: http://arxiv.org/abs/1806.03185

[97] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[98] F. Lluís, J. Pons, and X. Serra, "End-to-end music source separation: is it possible in the waveform domain?" *CoRR*, vol. abs/1810.12187, 2018. [Online]. Available: http://arxiv.org/abs/1810.12187

[99] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, p. 1256–1266, Aug 2019. [Online]. Available: http://dx.doi.org/10.1109/TASLP.2019.2915167

[100] I. Kavalerov, S. Wisdom, H. Erdogan, B. Patton, K. W. Wilson, J. L. Roux, and J. R. Hershey, "Universal sound separation," *CoRR*, vol. abs/1905.03330, 2019. [Online]. Available: http://arxiv.org/abs/1905.03330

[101] B. Kadioglu, M. Horgan, X. Liu, J. Pons, D. Darcy, and V. Kumar, "An empirical study of conv-tasnet," 2020.

[102] D. Samuel, A. Ganeshan, and J. Naradowsky, "Meta-learning extractors for music source separation," 2020.

[103] A. A. Nugraha, A. Liutkus, and E. Vincent, "Multichannel audio source separation with deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 9, pp. 1652–1664, 2016.

[104] S. I. Mimilakis, E. Cano, D. FitzGerald, K. Drossos, and G. Schuller, "Examining the perceptual effect of alternative objective functions for deep learning based music source separation," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, Oct 2018, pp. 679–683.

[105] H. Erdogan and T. Yoshioka, "Investigations on data augmentation and loss functions for deep learning based speech-background separation," in *19th Annual Conference of the International Speech Communication Association (Interspeech)*, 09 2018, pp. 3499–3503.

[106] A. Sahai, R. Weber, and B. McWilliams, "Spectrogram feature losses for music source separation," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[107] M. Kolbaek, Z.-H. Tan, S. H. Jensen, and J. Jensen, "On loss functions for supervised monaural time-domain speech enhancement," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, p. 825–838, 2020. [Online]. Available: http://dx.doi.org/10.1109/TASLP.2020.2968738

[108] J. R. Hershey, Z. Chen, J. L. Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," *CoRR*, vol. abs/1508.04306, 2015. [Online]. Available: http://arxiv.org/abs/1508.04306

[109] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering," 2020.

[110] Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, and N. Mesgarani, "Deep Clustering and Conventional Networks for Music Separation: Stronger Together," *arXiv e-prints*, p. arXiv:1611.06265, Nov 2016.

[111] T. Chan, T. Yeh, Z. Fan, H. Chen, L. Su, Y. Yang, and R. Jang, "Vocal activity informed singing voice separation with the ikala dataset," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 718–722.

[112] A. Liutkus, D. Fitzgerald, Z. Rafii, B. Pardo, and L. Daudet. (2013) Kernel additive modelling for source separation - ccmixter vocal separation database. [Online]. Available: https://members.loria.fr/ALiutkus/kam/

[113] R. C. L. Bittner, J. Wilkins, H. Yip, and J. P. Bello, "Medleydb 2.0: New data and a system for sustainable data collection," in *International Conference on Music Information Retrieval (ISMIR-16)*, 2016.

[114] A. Liutkus, F.-R. Stöter, Z. Rafii, D. Kitamura, B. Rivet, N. Ito, N. Ono, and J. Fontecave, "The 2016 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015*, 2016.

[115] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2018.

[116] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019.

[117] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *CoRR*, vol. abs/1712.04621, 2017. [Online]. Available: http://arxiv.org/abs/1712.04621

[118] R. Gontijo-Lopes, S. J. Smullin, E. D. Cubuk, and E. Dyer, "Affinity and diversity: Quantifying mechanisms of data augmentation," 2020.

[119] K. Choi, G. Fazekas, M. B. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," *CoRR*, vol. abs/1703.09179, 2017. [Online]. Available: http://arxiv.org/abs/1703.09179

[120] Q. Yang, Y. Zhang, W. Dai, and S. J. Pan, *Transfer Learning*. Cambridge University Press, 2020.

[121] M. Olson, A. Wyner, and R. Berk, "Modern neural networks generalize on small data sets," in *Advances in Neural Information Processing Systems*, 2018, pp. 3619–3628.

[122] W. Roth, G. Schindler, M. Zöhrer, L. Pfeifenberger, R. Peharz, S. Tschiatschek, H. Fröning, F. Pernkopf, and Z. Ghahramani, "Resource-efficient neural networks for embedded systems," *arXiv preprint arXiv:2001.03048*, 2020.

[123] F. Pernkopf, W. Roth, M. Zöhrer, L. Pfeifenberger, G. Schindler, H. Fröning, S. Tschiatschek, R. Peharz, M. Mattina, and Z. Ghahramani, "Efficient and robust machine learning for real-world systems," *CoRR*, vol. abs/1812.02240, 2018. [Online]. Available: http://arxiv.org/abs/1812.02240

[124] P. Chandna, M. Miron, J. Janer, and E. Gómez, "Monoaural audio source separation using deep convolutional neural networks," in *Lecture Notes in Computer Science*, vol. 10169, 02 2017, pp. 258–266.

[125] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, M. Hasan, B. C. V. Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *CoRR*, vol. abs/1803.01164, 2018. [Online]. Available: http://arxiv.org/abs/1803.01164

[126] M. Telgarsky, "Benefits of depth in neural networks," *arXiv preprint arXiv:1602.04485*, 2016.

[127] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[128] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[129] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[130] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[131] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research." in *ISMIR*, vol. 14, 2014, pp. 155–160.

[132] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[133] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017.

[134] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 1, p. 281–305, Feb. 2012.

[135] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," 2017.

[136] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," 2018.

[137] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[138] S. Jastrzebski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey, "Three factors influencing minima in sgd," *arXiv preprint arXiv:1711.04623*, 2017.

[139] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[140] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" in *Advances in Neural Information Processing Systems*, 2018, pp. 2483–2493.

[141] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2682–2690.

[142] T. Van Laarhoven, "L2 regularization versus batch and weight normalization," *arXiv preprint arXiv:1706.05350*, 2017.

[143] E. Hoffer, R. Banner, I. Golan, and D. Soudry, "Norm matters: efficient and accurate normalization schemes in deep networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 2160–2170.

[144] C. Summers and M. J. Dinneen, "Four things everyone should know to improve batch normalization," *arXiv preprint arXiv:1906.03548*, 2019.

[145] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[146] Y. Wang, A. Narayanan, and D. Wang, "On training targets for supervised speech separation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 22, no. 12, pp. 1849–1858, 2014.

[147] N. Takahashi, P. Agrawal, N. Goswami, and Y. Mitsufuji, "Phasenet: Discretized phase modeling with deep neural networks for audio source separation." in *Interspeech*, 2018, pp. 2713–2717.

[148] J. Le Roux, G. Wichern, S. Watanabe, A. Sarroff, and J. R. Hershey, "Phasebook and friends: Leveraging discrete representations for source separation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 370–382, 2019.

[149] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 708–712.

[150] V. S. Kadandale, J. F. Montesinos, G. Haro, and E. Gómez, "Multi-task u-net for music source separation," 2020.

[151] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[152] N. Perraudin, P. Balazs, and P. L. Søndergaard, "A fast griffin-lim algorithm," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, pp. 1–4.

[153] Y. Masuyama, K. Yatabe, Y. Koizumi, Y. Oikawa, and N. Harada, "Deep griffin–lim iteration," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 61–65.

[154] Z. Pruša and P. L. Søndergaard, "Real-time spectrogram inversion using phase gradient heap integration," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, 2016, pp. 17–21.

[155] D. Williamson, Y. Wang, and D. Wang, "Complex ratio masking for monaural speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, pp. 1–1, 01 2015.

[156] Y. Lee, C. Wang, S. Wang, J. Wang, and C. Wu, "Fully complex deep neural network for phase-incorporating monaural source separation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 281–285.

[157] H.-S. Choi, J.-H. Kim, J. Huh, A. Kim, J.-W. Ha, and K. Lee, "Phase-aware speech enhancement with deep complex u-net," *CoRR*, vol. abs/1903.03107, 2019, withdrawn. [Online]. Available: http://arxiv.org/abs/1903.03107

[158] L. Pfeifenberger, M. Zöhrer, and F. Pernkopf, "Deep complex-valued neural beamformers," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. IEEE, 2019, pp. 2902–2906. [Online]. Available: https://doi.org/10.1109/ICASSP.2019.8683517

[159] L. Drude, B. Raj, and R. Haeb-Umbach, "On the appropriateness of complex-valued neural networks for speech enhancement," in *INTERSPEECH 2016, San Francisco, USA*, 2016.

[160] J. Muth, S. Uhlich, N. Perraudin, T. Kemp, F. Cardinaux, and Y. Mitsufuji, "Improving dnn-based music source separation using phase features," *arXiv preprint arXiv:1807.02710*, 2018.

[161] K. W. Cheuk, H. Anderson, K. Agres, and D. Herremans, "nnaudio: An on-the-fly gpu audio to spectrogram conversion toolbox using 1d convolution neural networks," *arXiv preprint arXiv:1912.12055*, 2019.

[162] Z.-Q. Wang, J. L. Roux, and J. R. Hershey, "Alternative objective functions for deep clustering," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 686–690.

[163] P. Seetharaman, G. Wichern, S. Venkataramani, and J. L. Roux, "Class-conditional embeddings for music source separation," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 301–305.

[164] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 246–250.

[165] R. Kumar, Y. Luo, and N. Mesgarani, "Music source activity detection and separation using deep attractor network." in *Interspeech*, 2018, pp. 347–351.

[166] H. Lin, Z. Shi, and Z. Zou, "Maritime semantic labeling of optical remote sensing images with multi-scale fully convolutional network," *Remote sensing*, vol. 9, no. 5, p. 480, 2017.

[167] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[168] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," Jan. 2020, 0.7.2. [Online]. Available: https://doi.org/10.5281/zenodo.3606573

[169] M. Slaney, "Auditory toolbox," *Interval Research Corporation, Tech. Rep*, vol. 10, no. 1998, 1998.

[170] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger, "Memory-efficient implementation of densenets," *arXiv preprint arXiv:1707.06990*, 2017.

[171] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 1963–1972. [Online]. Available: http://papers.nips.cc/paper/7466-pelee-a-real-time-object-detection-system-on-mobile-devices.pdf

[172] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[173] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 116–131.

[174] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2752–2761.

[175] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *arXiv preprint arXiv:1908.00709*, 2019.

[176] E. Real, C. Liang, D. R. So, and Q. V. Le, "Automl-zero: Evolving machine learning algorithms from scratch," *arXiv preprint arXiv:2003.03384*, 2020.

[177] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[178] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019. [Online]. Available: http://arxiv.org/abs/1905.11946

[179] Y. Guo, "A survey on methods and theories of quantized neural networks," *arXiv preprint arXiv:1808.04752*, 2018.

[180] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognition*, p. 107281, 2020.

[181] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Training pruned neural networks," *CoRR*, vol. abs/1803.03635, 2018. [Online]. Available: http://arxiv.org/abs/1803.03635

[182] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[183] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," *arXiv preprint arXiv:1902.09574*, 2019.

[184] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *arXiv preprint arXiv:2003.03033*, 2020.

[185] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[186] K. Paupamah, S. James, and R. Klein, "Quantisation and pruning for neural network compression and regularisation," *arXiv preprint arXiv:2001.04850*, 2020.

[187] S. Ye, T. Zhang, K. Zhang, J. Li, J. Xie, Y. Liang, S. Liu, X. Lin, and Y. Wang, "A unified framework of DNN weight pruning and weight clustering/quantization using ADMM," *CoRR*, vol. abs/1811.01907, 2018. [Online]. Available: http://arxiv.org/abs/1811.01907

[188] S. Ye, X. Feng, T. Zhang, X. Ma, S. Lin, Z. Li, K. Xu, W. Wen, S. Liu, J. Tang, M. Fardad, X. Lin, Y. Liu, and Y. Wang, "Progressive dnn compression: A key to achieve ultra-high weight pruning and quantization rates using admm," 2019.

[189] H. Yang, S. Gui, Y. Zhu, and J. Liu, "Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach," 2019.

[190] X. Ma, S. Lin, S. Ye, Z. He, L. Zhang, G. Yuan, S. Huat Tan, Z. Li, D. Fan, X. Qian *et al.*, "Non-structured dnn weight pruning–is it beneficial in any platform?" *arXiv*, pp. arXiv–1907, 2019.

[191] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," 2014.

[192] G. Schindler, W. Roth, F. Pernkopf, and H. Froening, "Parameterized structured pruning for deep neural networks," *arXiv preprint arXiv:1906.05180*, 2019.

[193] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013.

[194] J. Bai, F. Lu, K. Zhang *et al.*, "Onnx: Open neural network exchange," https://onnx.ai/, 2020.

[195] Microsoft, https://microsoft.github.io/onnxruntime/, 2020.

[196] Intel, https://github.com/oneapi-src/oneDNN, 2020.

[197] Z. Pruša and P. Rajmic, "Toward high-quality real-time signal reconstruction from stft magnitude," *IEEE signal processing letters*, vol. 24, no. 6, pp. 892–896, 2017.

[198] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates," 2019.

[199] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *arXiv preprint arXiv:2002.05651*, 2020.

[200] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin, "Hardnet: A low memory traffic network," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.